a SD11
.A42
Reserve

United States Department of Agriculture

Forest Service

Rocky Mountain Forest and Range Experiment Station

Fort Collins, Colorado 80526

General Technical Report RM-235

Operations Guide for FORPLAN on Microcomputers (Release 14.2)

Brian M. Kent
Michael Bevers
Katherine E. Sleavin
James P. Merzenich
Jill Heiner
Matt Turner
Sarah B. Hall

Acknowledgments

The authors would like to thank the many people who helped contribute to this operations guide. In particular, comments and suggestions made by Robert Lee of Region 2, Sarah Crim, and Richard Phillips of Region 6 were especially helpful.

USDA Forest Service
General Technical Report RM-235

September 1993

Operations Guide for FORPLAN on Microcomputers (Release 14.2)

Brian M. Kent, Research Forester
Rocky Mountain Forest and Range Experiment Station[1]

Michael Bevers, Research Forester
Rocky Mountain Forest and Range Experiment Station[1]

Katherine E. Sleavin, Operations Research Analyst
Washington Office LMP Systems Section, Fort Collins, CO

James P. Merzenich, Operations Research Analyst
Region 6 Regional Office, Portland, OR

Jill Heiner, Systems Analyst
Management Assistance Corporation, Fort Collins, CO

Matt Turner, Planner
Payette National Forest, McCall, ID

Sarah B. Hall, Programmer/Analyst
Washington Office LMP Systems Section, Fort Collins, CO

[1]Headquarters is in Fort Collins, in cooperation with Colorado State University.

Contents

Operations Guide for FORPLAN on Microcomputers (Release 14.2)

Brian M. Kent, Michael Bevers, Katherine E. Sleavin,
James P. Merzenich, Jill Heiner, Matt Turner, and Sarah B. Hall

CHAPTER 1. INTRODUCTION

A. PURPOSE AND SCOPE OF THE GUIDE

This operations guide provides information on installing and using FORPLAN Version 2, Release 14.2, on MS DOS[2] microcomputers and gives troubleshooting information relating to operating FORPLAN on these computers. This guide includes information on 1) the hardware and software needed to run FORPLAN on a micro platform; 2) the steps necessary to obtain and execute FORPLAN; and 3) information on software or utilities developed to date that deal with FORPLAN analysis in a microcomputing environment. This guide does not explain what FORPLAN is, how to prepare data input, or how to address the issues relating to planning analysis. A list of publications and documents that discuss these items are given later in this chapter. It is assumed that the reader has a basic knowledge of the MS DOS operating system; hence DOS commands will not be explained in detail. It is also assumed that the reader has a fundamental understanding of FORPLAN terminology, linear programming (LP), and the process involved in generating and solving FORPLAN matrices.

This operations guide is presented in two forms. The first is intended primarily for users within the USDA Forest Service and resides in electronic format on the agency Data General (DG) office network. It is a "living document" in that it will be updated and enhanced as new information becomes available or new technology is incorporated. Many of the current software systems that support FORPLAN are continuously updated or replaced. The second form is printed as a General Technical Report, produced by the Rocky Mountain Forest and Range Experiment Station and suitable for more widespread distribution.

Chapter 1 contains an introduction to FORPLAN and presents a historical perspective and outline of the development and evolution of the system, culminating in the current release 14.2 of Version 2. It also provides information about contacts for additional assistance and offers other literature sources that users can refer to for background information.

Chapter 2 describes the hardware environment necessary to execute FORPLAN on microcomputers. It presents information on the location and procedure to obtain copies of the software, and discusses the method of how to set up FORPLAN on your system. The numerous FORPLAN programs are discussed along with detailed instructions on their interaction. File management and test data sets are also discussed. Finally, the chapter outlines differences between micro FORPLAN Release 13 (Kent et al. 1992) and Release 14.2.

Chapter 3 provides an introduction to the generation of batch files using the BATGEN software.

Chapter 4 details procedures to check FORPLAN input data files for errors and to generate a matrix.

Chapter 5 discusses procedures to solve an LP once the matrix generation is complete. Both of the commercial LP model solvers that are supported, LINDO and C-WHIZ, are discussed.

Chapter 6 explains generating FORPLAN reports using PRESENT, Paradox, Oracle, and the standard FORPLAN report writing software.

Numerous appendices contain information on file names, additional report generation software, and supplementary programs.

[2]The use of trade and company names is for the benefit of the reader; such use does not constitute an official endorsement or approval of any service or product by the U.S. Department of Agriculture to the exclusion of others that may be suitable.

B. GENERAL DESCRIPTION OF FORPLAN VERSION 2

FORPLAN Version 2 is an LP-based forest planning system used to optimize both the land allocation and activity and output scheduling for a forest over a specified planning horizon. It includes a matrix generator and a report writer. Commercial LP packages are used to solve the LP problems generated by FORPLAN. The matrix generator reads and interprets the data set and creates the rows and columns for the LP software to solve. The report writer interprets the LP solution and produces a series of user-specified reports and/or a flat file for use with other reporting software.

Enhancements and corrections to the FORPLAN Version 2 code are facilitated through a numbered release system, supported by the USDA Forest Service Land Management Planning Systems Section in Fort Collins. A new release of FORPLAN is distributed when one or more major enhancements have been incorporated in the system, or when a significant number of errors are corrected. To date, there have been 14 releases of the Version 2 code. A modified form of Release 13 and two editions of Release 14 (Release 14.1 and the subject of this guide, Release 14.2) have been converted to operate on microcomputers.

C. HISTORICAL PERSPECTIVE

FORPLAN Version 1 was first released in 1980 and was developed in an effort to merge timber and land management planning (Iverson and Alston 1986). Initial development was conducted by Dr. K.N. Johnson while he worked at Utah State University through a cooperative effort by the Department of Forestry and Computer Center at Utah State with Timber Management and Land Management Planning of the USDA Forest Service, Fort Collins.

FORPLAN Version 1 was developed to address the issues of integrated resource planning on national forests and became the "required primary analysis tool" for use in forest planning. It was the first LP-based system to recognize land management activities and outputs other than those directly associated with timber harvesting. FORPLAN Version 2 was introduced in 1983 and contained significant improvements over Version 1 (Johnson 1986; Johnson et al. 1986). For many years both Version 1 and Version 2 operated only on the UNISYS (UNIVAC) 1100/92 computer located at the USDA Fort Collins Computer Center (FCCC).

In 1987, Johnson developed a microcomputer version of FORPLAN known as "PC FORPLAN" (Kent et al. 1992). It required 640 KB of RAM, a hard disk with at least 10 MB of storage, and a floating-point coprocessor (8087 or 80287). This attempt was the first to move the FORPLAN processing environment from a mainframe computer to a more user-friendly environment. PC FORPLAN could use either LINDO or MINOS as the LP optimizer. This first attempt at moving FORPLAN off of mainframes was only partially successful because there were limitations on the size of problems that could be formulated and solved. These limitations arose because PC FORPLAN executables could only address the 640 KB of RAM that MS DOS recognized — the technology to do more than this did not exist in 1987.

In 1989 scientists at the Rocky Mountain Forest and Range Experiment Station developed another microcomputer version of FORPLAN called RMS FORPLAN (Kent et al. 1992). The source code used in RMS FORPLAN was also downloaded from the UNISYS and was based on a slightly enhanced form of Release 13. RMS FORPLAN (hereafter, Release 13) retained all of the capability of the main-frame version/release because of the utilization of technology developed for 80386 and newer Intel CPUs. This technology, known as DOS extenders, enables programs such as FORPLAN to access as much RAM as required even in excess of 640 KB; the only limiting factor is the amount installed on a given machine. Release 13 was tested extensively in Region 6 (Pacific Northwest Region of the National Forest System) during 1989-1990 and was instrumental in the development of forest plans on a number of forests in that region (Ager et al. 1991).

Release 14.0 of Version 2 FORPLAN became available on the UNISYS main-frame in April, 1991. This release featured two major enhancements to the FORPLAN software — the capability to formulate Model 2-type scheduling models and the capability to create a new type of flat file report to load into relational database software. The Model 2 activity scheduling capability is generalized beyond that outlined in Johnson & Schuerman (1977) and is an improved version of the generalizations described in Johnson et al. (1986) and Johnson and Stuart (1987). The Model 2 enhancements also led to extensive redesign of portions of the FORPLAN source code.

Later in 1991, the Rocky Mountain Station, in cooperation with the WO Land Management Planning Systems Section, downloaded Release 14 to the MSDOS environment. This second download afforded the opportunity to take advantage of substantial improvements in DOS-based FORTRAN compilers, DOS extenders, make utilities, and system management utilities that occurred since 1989. Also, this project provided a useful environment for rigorous FORPLAN system testing and modification. All errors reported with Release 13 (and Release 14.0 on the mainframe) were corrected. Additionally, several minor changes were made to the system over Release 13 to improve its ease of use in areas such as file management. A summary of the more important changes is given in Chapter 2. This code was denoted as Release 14.1. The most recent changes occurred in the spring of 1992 when a slightly enhanced version of Release 14.1 (Release 14.2, hereafter FORPLAN) was made available to users.

FORPLAN Release 14.2 requires an IBM-compatible PC with an Intel 80386 CPU and a math coprocessor, or an Intel 80486 CPU. Two versions of the system exist, one for an Intel coprocessor (can be either an 80387 or built into the CPU in the case of an 80486), and one for a Weitek Abacus coprocessor. A minimum of 4 MB of RAM and a suitable hard drive are required, and a minimum of 100 MB of free hard drive space should be available. The amount of hard drive space required depends on how you manage your file system and on the size of the problem (LP model) being formulated. See Chapter 5 for more discussion of this point.

D. ASSISTANCE CONTACTS

A user's guide (LMP 1992) documenting the preparation of input data and yield files is available. The user's guide is divided into sections corresponding to the data input sections. A complete user's guide can be obtained from the Forest Service Land Management Planning (LMP) Information Center on the DG, or by contacting the Systems Section (see phone number and address below).

The LMP Systems Section office in Fort Collins, Colorado, provides telephone user support for the FORPLAN Version 2 system to USDA Forest Service employees. Problems may be encountered when executing FORPLAN, or any of the related programs, on any type of computing platform. These problems tend to fall into one of two categories: problems in formulating a model or understanding the results, and problems in the system or program. The first type of problem can be solved by consulting the numerous manuals and publications on FORPLAN, consulting your Regional Analyst, or dialing the hotline:

FORPLAN HOTLINE
(303) 498-1833

Mail can be addressed to: Land Management Planning
Attn: FORPLAN
USDA Forest Service
3825 E. Mulberry St.
Fort Collins, CO 80524

Problems involving FORPLAN program errors should be reported directly to the LMP Systems Section. To date, FORPLAN has been run on many brands of machines with minimal incompatibilities. Incompatibilities that did occur appear to have been caused by using versions of DOS previous to 3.31 or from a peculiar ROM BIOS. If FORPLAN does not execute on your system, check that your system meets the minimum hardware specifications stated in Chapter 2. If a problem arises from incompatibilities between your system and FORPLAN, the Systems Section's staff will work with you to resolve the difficulty.

USDA Forest Service employees who wish to access current information concerning FORPLAN can have their names put on the ANALYSTNET mailing network on the Data General (DG) system. This network serves as a means of communication among all analysts and planners in the Forest Service. The Forest Service also supports an Information Center Service for planning tools, of which FORPLAN is one. If you want your name to be placed on the network, or if you would like information on the Information Center, contact:

K.SLEAVIN:W04A

A tutorial has been developed to assist users of FORPLAN with the various features of the system. This tutorial consists of a series of 12 data sets, yield files, questions, and answers. The tutorial resides in the Forest Service LMP Information Center on the DG. Data files can be transferred from the DG to your microcomputer.

E. SUPPORT LITERATURE

This operations guide does not describe the FORPLAN system itself in any detail; it considers only the mechanics of operating FORPLAN on a microcomputing platform. For more information on FORPLAN internal operations and planning analysis, we suggest the following:

Kent (1989) provides a detailed explanation of the aspects of linear programming that must be understood to use it effectively in the development of forest planning models.

Iverson and Alston (1986) describe the historical development of systems that led to the use of FORPLAN and trace the shift from traditional single-resource scheduling to interdisciplinary, integrated, multi-resource management planning.

Johnson et al. (1986) provide an overview of FORPLAN Version 2 and explain how it addresses forest planning problems in terms of the system's development, purpose, and capability.

The user's guide for Version 2 (LMP 1992) contains the logic, rules, limits and formats for model formulation and data preparation. This document can be obtained electronically from the LMP Information Center on WO4A, or a hard copy can be requested by contacting the LMP Systems Section.

Johnson and Stuart (1987) explain the mathematical structures that can be represented in FORPLAN Version 2. It is optional reading for the analyst building a model, but may be helpful in developing a deeper understanding of the FORPLAN system.

During 1986, the agency conducted two workshops relating to FORPLAN and its use, and the proceedings from each contain useful information (Bailey 1986; Hoekstra et al. 1987).

CHAPTER 2. INSTALLING FORPLAN ON MICROCOMPUTERS

A. HARDWARE SPECIFICATIONS

The minimum hardware specifications for running FORPLAN on a microcomputer are as follows:

- — An IBM compatible 80386 or 80486 machine
- — A minimum of 4 MB of RAM
- — A hard disk with at least 100 MB free space.

A math coprocessor is required to run FORPLAN or any of the LP solver software. If a given machine has an 80386 CPU, then either the Intel 80387 or the Weitek is suitable. If the machine has an 80486 CPU, then the equivalent of an 80387 is built in and no additional coprocessor is required. A Weitek may be utilized although the performance gain is not as significant as that realized on an 80386 machine.

Following are some additional points to consider after you have installed FORPLAN on your machine:

1) Some of the executable programs that make up FORPLAN create numerous scratch files that use considerable disk storage during run time. These scratch files will be deleted after FORPLAN has executed, but disk space is used and accessed during run time. For example, it takes twice as much disk storage space to *create* a flat file as it does to *store* it after creation.

2) The matrix generation process in FORPLAN may require substantial hard drive space when creating and building the LP matrix chapters. Be aware of the number of nonzero coefficients in your model. A model with 1 million nonzero coefficients will require approximately 40-45 MB of hard disk space just to store the matrix chapters. You should have two and one-half to three times the amount of storage required for your model available as free storage because two copies of your LP data will reside on the hard drive to accommodate LINDO or C-WHIZ. See Chapter 5 for more details on this issue.

3) If you execute FORPLAN without adequate hard disk space, your system will lock up. For example, typical National Forest models developed in Region 6 require from 60 to 150 MB for storage. A warm boot will usually be sufficient to restore the system, but your run will need to be restarted once the space problems have been resolved.

B. FORPLAN SOFTWARE

FORPLAN software contains many separate executable programs, sample data sets, sample batch files, and a batch file creation program. All of the programs can be obtained free of charge for Forest Service employees from the WO-LMP support staff in Fort Collins. Copies can be RISed via the DG and transferred to your microcomputer. All programs and other necessary files are contained in four large self-extracting ZIP files. The ZIP files are located in

HOST:	W04A
STAFF:	LMP
DRAWER:	FORPLAN
FOLDER:	V2R14_INTEL or V2R14_WEITEK

The V2R14_INTEL folder contains the programs and files needed if you use the Intel math coprocessor and V2R14_WEITEK contains programs and files needed if you use the Weitek math coprocessor.

Two files can be RISed from each folder. From V2R14_INTEL, you may RIS FPEXE_I.EXE and ZONE_I.EXE. From V2R14_WEITEK, you may RIS FPEXE_W.EXE and ZONE_W.EXE. The FPEXE file is approximately 1.6 MB and contains FORPLAN executables, batch files, utilities, and a sample data set. The ZONE file is approximately 0.9 MB and contains executables, batch files and a sample data set all specific to zone formulations. If your model does not use a zone formulation, you only need to RIS the appropriate FPEXE file.

These are large files that will require several thousand blocks of DG storage and may take a while to transfer. At a 9600 baud rate FPEXE_I.EXE requires approximately 35 minutes to transfer, and ZONE_I.EXE requires 15 minutes to transfer. We suggest that you perform the file transfer during off-peak hours. You must have owner, edit, or append access to a staff drawer on your local DG system to retrieve the file(s). Carry out the following steps to retrieve the FORPLAN files:

1) From CEO get into IS (by choosing the following menu items):
   - 7. Utilities
     - 6. User Applications
       - 1. Run
         - 2. IS
           - 1. Standard
2) Choose option 3 (utilities) from the IS Main Menu.
3) Choose option 6 (transfer) from the utilities functions menu.
4) Choose option 1 (information transfer).
5) Choose option 2 (retrieve).
6) Choose level 2 (staff).
7) Enter the location where you want to store the retrieved file(s). This location must already exist on your DG host.
8) Enter the above location and file information when prompted for the location of the file(s) to be retrieved.
9) Only one file can be retrieved at a time.

Once the file resides on your DG, transfer it to your microcomputer using CEO Connect by carrying out the following instructions:

1) Sign onto the DG as usual.
2) From the CEO main menu get into IS (see instructions above).
3) Get into the staff-drawer-folder of the file to be transferred.
4) Enter CTRL-BRK to enter the CEO Connect main menu.
5) Choose option 5 (retrieve a file).
6) Enter the full host path name of the file to be retrieved.
7) Enter the DOS path name of the location to store the transferred file.

You may use the mask characters * and ? for step 9) and the % character in step 10), to set up the transfer of more than one file to your PC. If you do this, all files that match the file specification will be transferred in sequence. If this process is executed within IS and within the staff-drawer-folder location of the files to be transferred, only the file specification need be entered in step 9. As an alternative, transfer may be executed from CEO/CLI using the full pathname without entering IS. After file transfer is complete, it is sometimes necessary to enter CTRL-D three times to get the prompt back on your computer screen.

Typically it will require two or more hours to transfer FORPLAN from W04A to your PC. Also, the process will tie up both your DG profile and your PC during transfer.

Because all four files (FPEXE_I.EXE, FPEXE_W.EXE, ZONE_I.EXE, ZONE_W.EXE) are self-extracting ZIP files, you do not need any special software resident on your micro to unzip them. Simply type in the file name (i.e., FPEXE_I.EXE) and the file will unzip itself. When you unzip FPEXE_I.EXE, 20 new files that require approximately 3.8 MB of storage will be produced; doing the same for ZONE_I.EXE results in the creation of 14 files requiring approximately 2.1 MB of storage. A list and brief description of these files is given later in this chapter. A list of all the Weitek-related files created if you unzip FPEXE_W.EXE and ZONE_W.EXE is given in Appendix A. Note that you may delete the self-extracting ZIP files after you have unzipped them to save storage space.

CAUTION! You may get a system error when you attempt to unzip your file. The error will read "program too big to fit into memory" because there is insufficient RAM available. First try a warm boot of your system to clear any programs (such as DGCONFIG) that are still resident in memory. Try to unzip the file after the warm boot. If you still get the error message, then you have a resident program (such as Windows) that is taking up too much memory. You may have to unload one or more of these resident programs to create enough free memory.

FORPLAN Release 14.2 for micro computers is available to non-agency users from
Dr. Larry Davis
Department of Forest and Resource Management
University of California
Berkeley, CA 94702
(510)642-6489
Cost: $500.

C. SETTING UP FORPLAN ON YOUR SYSTEM

First, configure your system by incorporating the following settings in your CONFIG.SYS file:

FILES=40
BUFFERS=40

Remember to do a warm boot after you have changed the CONFIG.SYS file to have DOS recognize these new settings.

Second, assuming FORPLAN is installed on the C drive, set up your directories and files as follows:

1) Create a directory called **C:\FORPLAN** (or **C:\WFORPLAN** if you are using the Weitek version of the executables).
2) Copy all of the programs and files associated with FORPLAN into this directory, or transfer the necessary ZIP files to this directory and unzip them there.
3) Modify your AUTOEXEC.BAT file to incorporate **C:\FORPLAN** into the path statement. For example, **PATH C:\;C:\DOS;C:\FORPLAN**.
4) Create a directory on the C drive for each data set. Run each data set through FORPLAN from its own directory. This way, you can keep track of all files created by FORPLAN for each data set.
5) Use the BATGEN program described in Chapter 3 to create batch files (files with a .BAT extension) needed to run FORPLAN against your data. There are example .BAT files provided with the FORPLAN distribution package that are typical of files BATGEN will create but they may not be tailored to the organization of your system; it may be more convenient to build your own.

D. SUMMARY OF THE FORPLAN FILES

Following is a list of the files created when FPEXE_I.EXE is unzipped. The specific chapter (or chapters) that provides additional information on each file is given. The first three files are FORPLAN executable programs:

1)	MATPDO.EXE	Reads and checks the nonzone data portion of all data sets for errors. Prints data interpretation and information on errors to a file named MAT.FIL, and yield data interpretation and information on errors to YIELDTBL.FIL. (See Chapter 4)

2) MATGEN.EXE	Reads and checks all data sets for errors, and generates the LP data unless FORPLAN data errors are found. Prints data interpretation and error information to MAT.FIL, and yield data interpretation and information on errors to YIELDTBL.FIL. (See Chapter 4)
3) RPTWTR.EXE	Generates print file reports and two types of flat files. Uses the FORPLAN input data set, the yield data, and the LP solution results in SOL.FIL. Reports are printed in REPORT.FIL, the standard flat file is contained in FLAT.FIL, and the relational flat file is contained in RDBFLAT.FIL. The report writer will also produce YIELDTBL.FIL. (See Chapter 6)

Other miscellaneous executable programs are:

1) TIM.EXE	Writes the current day, month, time, and year, as determined by the computer system clock and calendar, to various log files (*.LOG). The times can be used to calculate wall clock times required to run various FORPLAN and LP solver programs on different data sets. (See Chapter 3)
2) LMPSFX.EXE	This LINDO preprocessor concatenates the LP matrix files (MATRX.*) into one data file (MPS.FIL), and replaces the embedded blanks in the row and column names with underscores. (See Chapter 5)
3) LSOLFX.EXE	This LINDO postprocessor edits the LINDO solution file (FILE.SOL) changing the underscores back to blanks and reformatting the solution information. This processing is necessary to ensure that the FORPLAN report writer can read the solution results (SOL.FIL). (See Chapters 5 and 6)
4) BEEPEND.EXE	This utility will "beep" in a high pitch to signal the successful completion of a batch file. Each batch file will execute BEEPEND upon successful completion of all steps. (See Chapter 3)
5) BEEPERR.EXE	This utility will "beep" in a low pitch to signal premature termination of batch file execution as a result of some type of error. (See Chapter 3)
6) BATGEN.EXE	This program allows the user to build custom batch files as an alternative to using the distributed batch files. The file BATGEN.HLP provides additional information. (See also Chapter 3)
7) ROLLOVER.EXE	This program works in conjunction with BATGEN and C-WHIZ when a rollover run is desired. (See Chapter 5)

The next three files relate to an example problem without allocation zones that is distributed with FORPLAN and is described in more detail later in this chapter:

1) DATA.SIX	This file contains the FORPLAN input data for the sample data set without zones. (See this Chapter and Chapter 3)
2) YIELD.SIX	This file contains the yield information for the sample data set without zones. (See this Chapter and Chapter 3)
3) LPINP.DAT	This file contains LP problem parameter information specific to data set 6. In general, LPINP.DAT is read by LINDO prior to attempting to solve the LP model at hand. (See Chapters 3 and 5)

Finally, there are seven example batch files and a help file that describes them:

1) PDO.BAT	This file executes MATPDO.EXE. Use this batch file to error check a data set without zones. (See Chapter 4)
2) GEN.BAT	This file executes MATGEN.EXE. Use this batch file to generate a matrix for a data set without zones. (See Chapter 4)
3) REPORT.BAT	This file executes the report writer (RPTWTR.EXE). Use this batch file to create traditional formatted reports, flat files, or relational flat files. (See Chapter 6)
4) LINDO.BAT	This file executes LMPSFX.EXE as a preprocessor to LINDO, executes LINDO using the file LPINP.DAT and your matrix as input into LINDO, and finally executes LSOLFX.EXE as a post processor to LINDO to prepare a solution file (SOL.FIL) in a format that the FORPLAN report writer can read. (See Chapter 5)
5) C-WHIZ12.BAT	This file will create a concatenated file called MPS.FIL to be used as input to the C-WHIZ LP solver. It will also execute C-WHIZ and the post-processor program SOLB2A which prepares the solution file (SOL.FIL) in a format that the FORPLAN report writer can read. (See Chapter 5)
6) ALL-L.BAT	This file will execute both the matrix generator (MATGEN.EXE) and the report writer (RPTWTR.EXE), for data sets without zones. It also executes the LINDO LP solver after generating the matrix. (See Chapter 3)
7) ALL-W.BAT	This file serves the same purpose as ALL-L.BAT except that C-WHIZ is executed rather than LINDO. (See Chapter 5)
8) BATFILES.HLP	This help file contains a brief description of the seven batch files.

When ZONE_I.EXE is unzipped, the following additional files will be created. They consist of programs, data sets, and batch files that are specific to problems with allocation zones. There are 5 FORPLAN executable programs:

1) MATPDOM1.EXE	Reads and checks the coordinated allocation choice (CAC) zone data portion of data sets with zones for errors. Prints data interpretation and information on errors to a file named MAT.FIL, and yield data interpretation and information on errors to a file named YIELDTBL.FIL. This program is always run prior to MATPDO.EXE. (See Chapter 4)
2) MATGENM1.EXE	Reads and checks CAC zone data sets for errors, and provides information that MATGEN.EXE needs to generate the zone portion of an LP matrix. Prints data interpretation and error information to MAT.FIL, and yield data interpretation and information on errors to YIELDTBL.FIL. This program is always run prior to MATGEN.EXE. (See Chapter 4)
3) RPTZ1.EXE RPTZ2.EXE RPTZ3.EXE	These 3 files convert a solution file (SOL.FIL) to a zone file (ZONE.FIL), which the report writer (RPTWTR.EXE) then uses to produce zone reports. (See Chapter 6)

The next two files relate to an example problem which is distributed with FORPLAN, contains zones, and is described in more detail later in this chapter:

1) DATA.CAC	This file contains the FORPLAN input data for the sample data set with zones. (See this Chapter and Chapters 4, 5, and 6)

2) YIELD.CAC	This file contains the yield informaiion for the sample data set with zones. (See this Chapter and Chapters 4, 5, and 6)

Finally there are six batch files:

1) ZONEPDO.BAT	This file executes MATPDOM1.EXE and MATPDO.EXE together. Use this batch file to error check a data set with zones. (See Chapter 4)
2) ZONEGEN.BAT	This file executes MATGENM1.EXE and MATGEN.EXE together. Use this batch file to generate a matrix for a data set with zones. (See Chapter 4)
3) ZONEFILE.BAT	This file executes RPTZ1-Z3.EXE. Use this batch file to convert a solution file (SOL.FIL) to a zone file (ZONE.FIL) before doing a zone report run. (See Chapter 6)
4) ZONERPT.BAT	This file executes the report writer (RPTWTR.EXE). Use this batch file to create zone reports for traditional formatted reports, for flat files, or for relational flat files using the zone file (ZONE.FIL) created by running ZONEFILE.BAT. (See Chapter 6)
5) ALLZON-L.BAT	This file will execute both the matrix generator (MATGENM1.EXE and MATGEN.EXE) and the FORPLAN report writer (RPTWTR.EXE) for data sets with zones. It also executes the LINDO LP solver after generating the matrix. (See Chapter 5)
6) ALLZON-W.BAT	This file serves the same purpose as ALLZON-L.BAT except that it executes the C-WHIZ LP solver rather than LINDO. (See Chapter 5)

NOTE: You may want to consider what editor to purchase or use with your FORPLAN files. The matrix file and report file are 132 characters wide, and some editors cannot view or edit files of this width. Also, some editors have limits on the size of file (500 KB) they can load. Many editors such as Brief, Qedit, and Norton Editor handle large files and are widely used. Norton Editor has an "outline mode" that works well with FORPLAN data. One can pop back and forth between outline mode, which displays only the FORPLAN data section headers, and normal mode, which displays every line of the data file.

E. FILE MANAGEMENT

Substantial hard disk space may be required to execute FORPLAN. Hence, file management is of the utmost importance to ensure adequate execution and storage space. We recommend the following conventions be used:

1) Make liberal use of the print control sections of the FORPLAN input data. The data section "Prescriptions to Be Printed" can be used to reduce the size of the matrix file (MAT.FIL). The data section "Report Information to Be Printed" allows you to select the reports to be printed to the report file (REPORT.FIL). The data section "Report Print Control" allows you to select the information on FORPLAN input data and LP reports to be printed to the report file. The Report Print Control information is input on the TITLE data card. You can suppress the printing of the basic model data interpretation, the constraint model data interpretation, the LP rows information, or the LP "other" columns information. Information of these data input sections are in the user's guide (LMP 1992).

2) It may be easier to create several small reports or flat files by rerunning the report writer software rather than making one run that creates a large report (REPORT.FIL), flat file (FLAT.FIL), or relational flat file (RBDFLAT.FIL).

3) Check the amount of available hard disk storage before starting any run. Clean up old files to ensure adequate disk space for FORPLAN executions.

4) Delete or back up files as appropriate to maintain adequate disk space. If you have limited hard disk space copy your extra data files, matrix files, LP files, and reports to a portable storage medium, or purchase a larger hard drive.

5) Organize your files and directories so that you can keep track of what you are doing.

6) Small prototype models should be developed first. Sequentially add the more complex features to the prototype, and test each new feature for errors.

We suggest that each data set be located in a separate subdirectory. When FORPLAN is executed, output is sent to files with generic names (MAT.FIL, REPORT.FIL, etc.). If data sets are not physically separated, subsequent executions of FORPLAN will write over previous output files.

F. TEST DATA SETS

Two data sets are included in the FORPLAN distribution package for testing purposes. Each data set consists of a data file and a yield file. We encourage you to run one or both of these data sets and check the output files, and verify that the family of FORPLAN programs is operating correctly on your system. The file names of the data sets are as follows:

```
1)  DATA.SIX        2)  DATA.CAC
    YIELD.SIX           YIELD.CAC
```

The SIX data set does not have zones, and the CAC data set does have zones. We will be using these two data sets as examples in subsequent sections of this guide.

The following steps assume you have installed FORPLAN on your system as described in Section C. The first step in running any data set is to create a separate directory. For example, for data set SIX follow these steps:

1) Get into the directory containing your FORPLAN programs by typing
   CD \FORPLAN

2) Make a subdirectory under the FORPLAN directory for data set SIX files by typing
   MD SIX

3) Get into this directory by typing
   CD SIX

4) Copy the two files for data set SIX (assuming they are in your root directory) into this subdirectory and rename them to conform with the file-naming convention assumed in the distributed .BAT files:

   COPY C:\DATA.SIX   DATA.FIL

   COPY C:\YIELD.SIX   YIELD.FIL

Note that we could leave the extension names unchanged if we intend to use the BATGEN program to generate batch files (See Chapter 3).

To run the CAC data set, follow steps 1 through 4 above and create a different subdirectory name in step 2, such as CAC. Copy any of the .BAT files described above into the SIX subdirectory and execute them using the sample data. Or use the BATGEN program if you have installed it to create your own batch files. An example of how to create batch files for the SIX data set is given in the next chapter.

G. DIFFERENCES BETWEEN RELEASE 13 AND RELEASE 14.2

In this section, we briefly describe some of the major differences between the original and most current releases of FORPLAN that run on microcomputers.

File Handling

All files of interest to the user are now explicitly named and opened within FORPLAN or on runstream command lines. All other files (scratch files and program communication files) are still given FORT.* default names and must be deleted via the DOS command line. Both the distributed batch files and those created by BATGEN perform all necessary file handling work.

Error Handling

Function of the STATUS.TMP file has been enhanced to provide robust error handling between FORPLAN programs. Its main purpose from program to program is to let the subsequent executables know when an error has occurred so that further processing will be suspended. Because commercial LP solvers are not sensitive to STATUS.TMP, an explicit DOS test and jump command is used in all batch files to avert optimization and reporting when matrix generation errors have occurred.

STATUS.TMP is also used as a flag to let the principal matrix generation program MATGEN (as well as MATPDO) know that MATGENM1 (or MATPDOM1) has been executed for zone formulations. Both the matrix generator and report writer also use the STATUS.TMP file as a waste basket for unwanted printout (a role taken by FORT.27 in Release 13, see Kent et al. 1992). In the event of an unusual execution problem, this excess print can sometimes be useful for diagnostic purposes. In general, STATUS.TMP should not exist at the beginning of a FORPLAN run. Table 1 shows how STATUS.TMP is used as a run progresses.

LP Model Solution

While LP solver software is not part of the FORPLAN system per se, it is intimately related to it. Release 14.2 supports two LP solution packages, 386 LINDO (as did Release 13), and C-WHIZ. While other LP solvers could be used to solve FORPLAN models, the BATGEN batch file creation program can only build batch files for LINDO and C-WHIZ. LINDO is available from LINDO Systems, Inc. and C-WHIZ is available from Ketron, Inc. For more information on these packages and their acquisition, see Chapter 5.

Pre-LP Processing

To reduce preprocessing time required to generate an MPS input file, Ketron programmers enhanced C-WHIZ with the ability to handle the embedded blanks in FORPLAN row and column names. Additionally, the matrix generator now writes the necessary MPS header cards directly into appropriate matrix files. As a result, the DOS Copy command is used to concatenate the nine FORPLAN matrix chapters in an MPS input file. Tests conducted to date have indicated significant reductions in the time required to produce this input file over what had been encountered using the old LINDO preprocessor distributed with Release 13.

Because 386 LINDO (hereafter, LINDO) still does not correctly process row and column names with embedded blanks, a preprocessor (LMPSFX.EXE) is still used to concatenate FORPLAN's matrix chapters and to convert embedded blanks to underscores. This program replaces earlier preprocessors that were distributed with Release 13. They are not compatible with Release 14.2 matrix files because of changes in the handling of the MPS header cards, and because of changes in the ordering of FORPLAN columns (see the next section).

Table 1.—Uses of STATUS.TMP during FORPLAN runs.

Program	Beginning of Execution	End of Execution
MATGENM1 (MATPDOM1)	— Catalog new STATUS.TMP	— Write final error count to it.
MATGEN (MATPDO)	— If it exists, check final M1 error count. (a) If M1 errors, abort execution. (b) If no M1 errors, set zone flags. — If it does not exist, catalog it. — Divert excess print to it.	— If errors, write final error count to it. — If no errors, delete it.
LP SOLVER	— Proceed with DOS batch file Command: IF EXIST STATUS.TMP GOTO: ERROR where :ERROR is a DOS label near the end of the batch file.	
RPTZ1	— Catalog new STATUS.TMP.	— If errors, write final error count to it. — If no errors, delete it.
RPTZ2 and RPTZ3	— If it exists, abort execution. — If it does not exist, catalog it.	— If errors, write final error count to it. — If no errors, delete it.
RPTWTR	— If it exists, abort execution. — If it does not exist, catalog it. — Divert excess print to it.	— If errors, write final error count to it. — If no errors, delete it.

Post-LP Processing

Post-LP processors are still needed to reformat solutions from both C-WHIZ and LINDO for FORPLAN's report writer. Ketron produced and distributes a program (SOLB2A) for this purpose with C-WHIZ. LSOLFX.EXE is used with LINDO solutions and re-converts underscores in row and column names back to embedded blanks as well as reformatting the output.

To simplify and speed up LP solution processing, the order of FORPLAN columns placed in the MPS input file has been changed for models with allocation zones. While this change means that Generalized Upper Bound (GUB) optimizations are no longer directly supported, decision variables in solution no longer have to be sorted prior to reporting. Of course, analysts wishing to exploit GUB optimization with LP solvers which offer that solution algorithm can still manipulate matrix files to that end.

Flat File Reporting

Flat file support has been changed to reflect the new relational database flat file capability in Release 14.2. Under the expectation that Data General's present package is no longer a targeted reporting system, further enhancements to the original or standard FORPLAN flat file capability are not anticipated. Consequently, HEADER.FIL and PRESENT.FIL have been dropped from Release 14.2. A header card for the final version of the original flat file is now written out directly to FLAT.FIL by the report writer.

The relational database flat file (RBDFLAT.FIL) is potentially made up of two new files. If users create a ZONE.FIL solution for zone models (as is necessary to create zone reports, see Chapter 6), the RPTZ3.EXE program also generates zone relational information in a file named RDBFLAT.ZON. This file, if present, is then concatenated with FORT.19 from the report writer to create a full RDBFLAT.FIL.

Finally, unlike Release 13, the standard flat file is not generated by Release 14.2 unless the user's data set specifically flags the report writer to create it. However, RDBFLAT.ZON will be created every time RPTZ3.EXE is run in anticipation of subsequent use with zone reports.

H. MULTITASKING

FORPLAN has been successfully run in a DOS multitasking environment using DESQVIEW (Quarterdeck, Inc.). In this environment, FORPLAN and either LINDO or C-WHIZ can run in background mode while you are using other applications. To set up a multitasking environment using DESQVIEW, follow these steps:

1) Select the "Change or Add Program" option from the DESQVIEW main menu.

2) Under "Standard Options" select the following:

Program Name:	FORPLAN
Keys to use on Open Menu:	FP
Memory size (in K):	128
Program:	(leave blank)
Parameters:	(leave blank)
Directory:	C:\FORPLAN
Options:	
Writes to screen:	Y
Displays graphics:	N
Virtualize text:	N
Use serial ports:	N
Requires floppy diskette:	N

3) Under "Advanced Options" select the following:

System memory:	0
Max. program memory in size:	600
Script buffer size:	1000
Max. expanded memory size:	4200
Text pages:	4
Graphics pages:	0
Initial mode:	(leave blank)
Interrupts:	00 + FF
Max. height:	25
Start height:	25
Start row:	0
Max width:	80
Start width:	80
Start column:	0

4) Under "Shared Program" select the following:

Pathname:	(leave blank)
Data:	(leave blank)
Close on exit:	(leave blank)
Allows close window command:	Y
Uses math coprocessor:	Y
Share CPU when foreground:	Y
Can be swapped out:	(leave blank)
Uses own colors:	Y
Runs in background:	Y
Keyboard conflict:	0
Share EGA when foreground Zoomed:	Y
Protection level:	0

CHAPTER 3. BUILDING BATCH EXECUTION FILES: THE BATGEN VERSION 1.1 PROGRAM

In the previous chapter we noted the presence of several batch (.BAT extension) files that contain runstreams that are designed to run various logical combinations of the FORPLAN programs. They also run, when appropriate, either of the supported LP solution software packages C-WHIZ (Version 1.4 or newer) or LINDO. We also noted the presence of the BATGEN program (Version 1.1 or V1.1), which is designed to generate batch files similar to those mentioned above, but tailored to each user's individual situation. For example, you can specify names of your choice for yield and data files and BATGEN will construct a batch file utilizing your file specifications. BATGEN also provides the capability to generate and "string together" up to 9 runstreams in a single batch file so that they will execute in sequence. This capability is an excellent way to execute multiple FORPLAN runs when your computer is unattended. BATGEN also provides the capability to generate batch files to do certain specialized LP solution runs when using C-WHIZ (i.e., advanced basis and rollover runs).

BATGEN was initially developed at the Rocky Mountain Station and has been enhanced and is now supported by the WO-LMP unit in Fort Collins. The rest of this chapter describes an example of the use of BATGEN. Support contacts for BATGEN at WO-LMP are the contacts given previously for the rest of the FORPLAN system. Non-agency users can obtain BATGEN from

Dr. Larry Davis
University of California at Berkeley
Dept. of Forest and Resource Management
Berkeley, CA 94702
(415)642-6489

Support is also available from the same source.

A. RUNNING BATGEN: AN EXAMPLE

First, incorporate the directory in which BATGEN resides in your path statement to facilitate execution. In the discussion that follows, we execute BATGEN directly from DOS. We use the SIX sample data set adopting the directory structure suggested in Chapter 2, and assuming that BATGEN is executed from within the SIX subdirectory. Figure 1 shows the BATGEN introductory screen. Note a menu with three choices, the first choice (Runstream Generator) having been selected. Before proceeding, the implications of choices 2 and 3 are considered briefly. If choice 2 (Information About this Program) is selected, a screen showing a brief summary of information pertaining to BATGEN and its support of C-WHIZ V1.4 or higher is given. Because all of that information is presented elsewhere in this guide, the screen will not be reproduced. If the third option (Quit) is chosen, the execution of BATGEN is terminated and the user is returned to DOS. Finally, note that you may exit from BATGEN at any time by entering **<CTL>C**.

By choosing the first option, Runstream Generator, we are asking BATGEN to create a batch file that will contain one or more runstreams. Given this choice, the next question asked by BATGEN is

Enter runstream filename (a ".BAT" extension is required and is assumed if not typed in)
(or hit <CR> for default of RUN.BAT): COMPLETE

Your response to this question determines the name of the batch file you are about to create. Here, we have selected the file name COMPLETE. Note that this file will be created in the directory from which BATGEN is being executed. If a file with the same name (in our example, COMPLETE.BAT) already exists in the directory, BATGEN will ask the following:

The file: COMPLETE.BAT already exists.
Do you want to overwrite it? (y or n): N

```
*************************************************************
*                       BATGEN                              *
*              FORPLAN BATCH FILE GENERATOR                 *
*                                                           *
*  Version 1.0   DEVELOPED by The Rocky Mountain Station    *
*                Fort Collins, CO (10/91)                   *
*  Version 1.1   ENHANCED by the Land Management Planning   *
*                Section (WO-LMP) Fort Collins, CO (1/92)   *
*  *** NOTE ***                                             *
*            Supports LP solvers LINDO and                  *
*            C-WHIZ version 1.4 (1-13-92)                   *
*************************************************************


 1. Runstream Generator
 2. Information about this program
 3. Quit

    Hit "<CTL> C" at any time to halt execution of program

    Enter number of choice:  1
```

Figure 1.—BATGEN introductory screen.

If you answer no, as we have here, BATGEN will repeat the original question asking for a batch file name.

The next screen contains a menu of possible run types that are logical combinations of FORPLAN and LP solver programs that can be executed in a single runstream (fig. 2). In this example, a complete run is selected. This selection means that BATGEN will construct a runstream that will execute the FORPLAN matrix generator to read and error check the FORPLAN input data and generate an LP matrix (if no data errors are found). The runstream will then execute either LINDO or C-WHIZ to solve the LP model. Finally, the runstream will execute the FORPLAN report writer to produce formatted reports and/or flat files based on the LP solution. This runstream will be located in COMPLETE.BAT. Information on the other run types will be presented in Chapters 4, 5, and 6.

```
                  RUNSTREAM GENERATOR

1. Complete Run
2. Matrix Generation
3. PDO
4. Matrix and LP
5. LP
6. LP and Report
7. Report
8. Zonefile Create

   Hit "<CTL> C" at any time to halt execution of program

   Enter number of choice:  1
```

Figure 2.—Run types supported by BATGEN.

The next question asked by BATGEN is

Are there zones in the model? (y or n): N

This question determines if allocation zones are present in your model. Refer to the FORPLAN user's guide (LMP 1992) for more information on allocation zones and coordinated allocation choices. We answered no because we are working with DATA.SIX and YIELD.SIX, which do not contain allocation zones. The next two questions ask you to identify the paths to and the names of the data and yield files to be processed:

Enter data file name, including drive and path,
(or hit <CR> for default of DATA.FIL): DATA.SIX

Enter yield file name, including drive and path,
(or hit <CR> for default of YIELD.FIL): YIELD.SIX

Note that our answers assume that these files reside in the same directory that COMPLETE.BAT does. If the file you specify for the data file does not exist in the directory that you specify, BATGEN will terminate your session and return you to the screen shown in figure 1. However, BATGEN does not check for the presence of the yield file you specify. At this point we have given BATGEN all the information it needs to construct the data editing and matrix generation portion of the runstream.

Because solution of the LP generated by FORPLAN is the next step, BATGEN's next prompt is

LP SOFTWARE
1. **LINDO**
2. **C-WHIZ**

Enter number of LP solver: 1

You have the choice between the two LP solvers, LINDO and C-WHIZ; we have selected LINDO. The next question relates to the selection of an objective function to optimize. BATGEN scans the objective function data in DATA.SIX and lists all objective functions for you to choose from. Here we select the first one:

OBJECTIVE FUNCTIONS
RETRIEVED FROM DATA.SIX
1. **OB1SOF_1 SOFTWOODS**
2. **OB2AAA15 TEST**

Enter number of choice: 1

Next, BATGEN will ask if you want to maximize or minimize:

Do you want to:
1. **Maximize**
2. **Minimize**

Enter number of choice: 1

In this case we have chosen maximization. The next two questions relate to the iteration limit we wish to set and whether we want LINDO to print iteration log information (showing how model solution is progressing) to the screen as it solves the model:

Enter max iterations (or hit <CR> for default of 100000): 4000
Do you want screen output? (enter a y or hit <CR> for "no"):

we select 4,000 iterations and the default for screen output.

Once the LP model is solved, the FORPLAN report writer is executed to generate formatted reports and flat files based on solution results. In this example, BATGEN already has all of the information it needs (i.e., data and yield file names) to build the report portion of the runstream, so the next question it asks is

Do you wish to add another runstream to this batchfile?
(You can create up to 9 runstreams) (enter a y or hit <CR> for "no"):

Here we select the default response because we are not interested in building another runstream. BATGEN responds with:

Finished created batch file....
and terminates its execution.

Figure 3 shows a listing of the runstream created by BATGEN in COMPLETE.BAT. Note that in addition to executing the FORPLAN programs MATGEN.EXE and RPTWTR.EXE and the LP solver LINDO87 the runstream does a lot of file-related housekeeping in terms of deleting and copying. Much of this work is done to eliminate possible file conflicts with the executable programs or to minimize the utilization of storage. Explanatory remarks are also incorporated pertaining to the file STATUS.TMP (see Chapter 2) and utilities that support LINDO (see Chapters 2 and 5). The final part of the runstream uses the presence (implying the run has aborted because of some error) or absence (implying normal termination) of the file STATUS.TMP to sound the appropriate tone.

```
echo off
Rem    Always start each run by deleting STATUS.TMP.  This file is used to
Rem    communicate error messages between programs.
echo on
del status.tmp
Rem    MATRIX GENERATOR WITHOUT ZONES, 80387 VERSION
del matrx.*
del fort.*
del mat.fil
del yieldtbl.fil
del mat.log
copy YIELD.SIX fort.4
copy DATA.SIX fort.5
tim.exe  >mat.log
matgen.exe <fort.5 >mat.fil
tim.exe >>mat.log
del fort.*
Rem    Stop Batch File If Matrix Errors
if exist status.tmp goto :error1

Rem    LP SOLVER (LINDO, 80387 VERSION)
del lp.fil
del mps.fil
del file.sol
del sol.fil
del lp.log
echo off
Rem    The following preprocessor for Lindo is required instead of a COPY
Rem    command due to Lindos inability to handle imbedded blanks in the
Rem    row & column names. This utility replaces MPSLND.EXE which is not
Rem    compatible with FORPLAN Release 14.1.
echo on
tim.exe >lp.log
lmpsfx.exe
echo off
```

```
Rem    Execute LINDO and the postprocessor LSOLFX.EXE to prepare a SOL.FIL.
Rem    Note that this batch file requires that problem specific info be in a
Rem    file named LPINP.DAT (analogous to the MPS.CR file in C-WHIZ). This
Rem    batch file also assumes that the LPINP.DAT file has identified the
Rem    input matrix file as MPS.FIL and the intermediate LINDO solution file as
Rem    FILE.SOL.  FILE.SOL is read by LSOLFX.EXE which is the postprocessor
Rem    that produces SOL.FIL for FORPLAN.
echo on
tim.exe >>lp.log
lindo87 <lpinp.dat >lp.fil
lsolfx.exe
tim.exe >>lp.log
del mps.fil
del file.sol

Rem    REPORT WRITER STANDARD REPORT, 80387 VERSION
echo off
Rem    The file RDBFLAT.FIL is the new relational database flat file.
echo on
del report.fil
del yieldtbl.fil
del flat.fil
del rdbflat.fil
del fort.*
del report.log
copy YIELD.SIX fort.4
copy DATA.SIX fort.5
copy sol.fil fort.11
tim.exe >report.log
rptwtr.exe <fort.5 >report.fil
tim.exe >>report.log
copy rdbflat.zon+fort.19 rdbflat.fil
copy fort.20+fort.21+fort.22+fort.23+fort .24+fort.25+fort.26+fort.27+fort.28+fort.30 flat.fil
del fort.*

Rem    Sound Appropriate Tone at End of Run
if exist status.tmp goto :error1
beepend
goto :end1
:error1
beeperr
:end1
echo End of Run #1
```

Figure 3—Listing of COMPLETE.BAT.

Note that BATGEN also created the file LPINP.DAT, which contains data input for LINDO (see Chapter 5). The contents of this file are shown in figure 4.

```
PAGE 0
WIDTH 132
TERSE
RMPS MPS.FIL
OB1SOF_1
MAX
DIVERT FILE.SOL
GO 2000
   2000
RPRI N T P R L U D :N = '%%%%%%%%'
CPRI N T P R L U D :N = '%%%%%%%%' .AND. P > 0
QUIT
```

Figure 4.—Listing of LPINP.DAT.

Figure 5 shows the contents of the directory containing COMPLETE.BAT after it is executed. The files MAT.FIL, YIELDTBL.FIL, and REPORT.FIL contain output from the matrix generator and report writer (see Chapters 4 and 6). LP.FIL contains output from LINDO and SOL.FIL contains the LP solution results formatted for the FORPLAN report writer (see Chapter 5). The MATRIX.* files contain the LP data created by the matrix generator (see Chapter 4) and RDBFLAT.FIL contains the relational data base flat file produced by the report writer (see Chapter 5).

```
Volume in drive E has no label
Volume Serial Number is 2649-13EE
Directory of C:\FORPLAN\SIX

.                    <DIR>          04-03-92      11:33a
. .                  <DIR>          04-03-92      11:33a
DATA       SIX           57811      08-22-89       1:44p
YIELD      SIX           18778      08-03-89      11:51a
COMPLETE   BAT            2236      04-08-92      12:55p
LP         LOG             147      04-08-92       2:23p
LPINP      DAT             178      04-08-92      12:49p
MAT        LOG              98      04-08-92       2:23p
MAT        FIL          221766      04-08-92       2:23p
LP         FIL             938      04-08-92       2:23p
MATRX      ROW              91      04-08-92       2:23p
MATRX      RX              608      04-08-92       2:23p
MATRX      RHS             157      04-08-92       2:23p
MATRX      RNG               8      04-08-92       2:23p
MATRX      OTH               9      04-08-92       2:23p
MATRX      CAC               0      04-08-92       2:22p
MATRX      BOT               8      04-08-92       2:23p
MATRX      BRX               8      04-08-92       2:23p
MATRX      BCA               0      04-08-92       2:22p
SOL        FIL            1085      04-08-92       2:23p
REPORT     LOG              98      04-08-92       2:23p
REPORT     FIL          124109      04-08-92       2:23p
RDBFLAT    FIL            8183      04-08-92       2:24p
YIELDTBL   FIL           60952      04-08-92       2:23p
           24 File(s)          221544448 bytes free
```

Figure 5.—Directory contents after executing COMPLETE.BAT.

A Note About .LOG Files

You may have noticed the presence of three files with a .LOG extension in figure 5. These files contain the starting and ending execution times for the matrix generator (MAT.LOG), LINDO (LP.LOG) and the report writer (REPORT.LOG). These times are taken from your system clock and calendar and are written to various .LOG files by the program TIM.EXE described in Chapter 2. The contents of MAT.LOG for our example are

```
Wed Oct 07 23:26:22 1992
Wed Oct 07 23:26:51 1992
```

This notation indicates that the matrix generator required 29 seconds to edit the input data and generate the LP model data for the SIX example. Any runstream you build using BATGEN will produce a .LOG file any time a FORPLAN program, LINDO, or C-WHIZ is executed so you can monitor the time it takes to build, solve, and report results for your models.

Multiple Runstreams in a .BAT File

It is possible to incorporate up to nine runstreams, each representing a different FORPLAN run, in a single .BAT file using BATGEN. Suppose after completing all questions pertaining to the first runstream, you answer as follows:

Do you wish to add another runstream to this batchfile?
(You can create up to 9 runstreams) (enter a y or hit <CR> for "no"): Y

You will then be asked two questions whose answers will establish the path to the directory where the next runstream is to be executed from:

Enter new drive plus a colon or hit <CR> to default to current drive: E:

Because we wish to execute our second runstream on the E drive, we answer as shown. The first executable command in the second runstream will accomplish this task. Then we are asked

Enter the path (including all backslashes) where you want to start runstream # 2, up to 30 characters can be used (EXAMPLE: \ROLL\ROLL1): \R14EXE\TOMCAC

The second command in this runstream will be

```
CD\R14EXE\TOMCAC
```

Thus the runstream that follows will be executed from the TOMCAC subdirectory on drive E. If you specify a subdirectory that does not exist, BATGEN will create it.

For the most part, the remaining questions (assuming another complete run) to be asked will not differ from those described for data set SIX. The one exception is the question pertaining to the location of the data and yield files that are to be used. This question is

Enter the data file name to use:
1. **DATA.FIL from current directory**
   (will be copied to E:\R14EXE\TOMCAC\DATA.FIL)
2. **Use E:\R14EXE\TOMCAC\DATA.FIL**
3. **Full pathname to another file**

Enter a 1, 2, or 3: 2

Note that you now have three choices for the original location of the data file. Here we have selected the second. This choice assumes DATA.FIL actually exists in the directory in question. If it does not, BATGEN will warn you that the file is not present and will prompt for another path and file name. BATGEN cannot proceed without a valid path and data file name. The yield file question is similar:

```
Enter the yield file name to use:
    1. YIELD.FIL from current directory
       (will be copied to E:\R14EXE\TOMCAC\YIELD.FIL)
    2. Use E:\R14EXE\TOMCAC\YIELD.FIL
    3. Full pathname to another file
    4. Do not use a YIELD.FIL in this run
Enter a 1, 2, 3 or 4:  2
```

Again, we have selected the second option and the comments about the data file apply here as well. If you wish to incorporate more than two runstreams, the process just outlined repeats itself.

This discussion completes our introduction to BATGEN showing some of its features and options. Additional capabilities will be explored in the next three chapters.

CHAPTER 4. GENERATING A MATRIX

There are three main steps in running FORPLAN: generating the LP model or matrix, solving or optimizing the model, and creating reports from the solution. This chapter covers the first step, generating the matrix, which has two phases: checking the data file for errors, and generating the matrix. We use the CAC sample data set as an example. The remaining steps will be discussed in Chapters 5 and 6.

A. CHECKING THE DATA FILE FOR ERRORS

When data sets are created they often contain many formatting and logical errors. Formatting errors include data in the wrong columns, incorrect codes, and missing data. Logical errors may include an illogical constraint formulation or incompatible relationships among different pieces of data. Before a matrix is generated it is necessary to check the data set for errors, correct the errors, and check again. Once the data set is error free, a matrix may be generated. Consult the user's guide (LMP 1992) for assistance in understanding error messages and correcting formatting and logical errors.

There are two ways to check a data file for errors: executing one of the provided batch files (PDO.BAT for data sets without zones or ZONEPDO.BAT for data set with zones), or running BATGEN to generate your own batch file as we do in this example.

Assuming that the BATGEN program is in your path, get into the subdirectory containing the CAC data and yield data. Here we assume that it is in

C:\FORPLAN\CAC

as suggested in Chapter 2. Upon executing BATGEN, you will first see the introductory screen shown in figure 1. Select option 1, Runstream Generator. The next question asks for the name of the batch file to be created

Enter runstream filename (a ".BAT" extension is required and is assumed if not typed in)
(or hit <CR> for default of RUN.BAT): ZONEPDO

Note that our answer results in the file ZONEPDO.BAT.

The next screen contains the list of possible run types (fig. 2) and we select **3. PDO** for data checking. The next question is

Are there zones in the model? (y or n): Y

We answer yes because we are using the CAC data set. Consequently, ZONEPDO.BAT will execute both MATPDOM1.EXE and MATPDO.EXE. (See Chapter 2 for more information on what these programs do.) The next two questions ask for the path and file names for the data and yield files:

Enter data file name, including drive and path,
(or hit <CR> for default of DATA.FIL): DATA.CAC
Enter yield file name, including drive and path,
(or hit <CR> for default of YIELD.FIL): YIELD.CAC

Note that our answers presume that the data and yield files are in the same directory as ZONEPDO.BAT. The final question is

Do you wish to add another runstream to this batchfile?
(You can create up to 9 runstreams) (enter a Y or hit <CR> for "no"):

As is discussed in Chapter 3, this question relates to the incorporation of more than one runstream in a given batch file. Because we wish to create only one runstream, we hit **<CR>** and BATGEN terminates execution with:

Finished creating batch file....

The contents of ZONEPDO.BAT are straightforward, especially if you have read and understood the information given in Chapter 3 on COMPLETE.BAT. Note that both MATPDOM1.EXE and MATPDO.EXE are run.

After you execute ZONEPDO.BAT, you will have the following new files (sizes in bytes in parentheses) in your subdirectory:

MAT.FIL (289,106)	This file contains FORPLAN's interpretation of the input data, exclusive of the yield data, for any error checking or matrix generation run.
YIELDTBL.FIL (112,542)	This file contains FORPLAN's interpretation of the yield data, as well as information on errors in the yield data, if any.
MAT.LOG (81)	This file records the starting and ending date and time information (according to your computer calendar and clock) for execution of the print data only software.

Look over each of these new files and check for errors. The matrix print file (MAT.FIL) contains the FORPLAN data edit program's interpretation of the input data (DATA.CAC). This file also includes information on any errors the FORPLAN program finds in the data set, and the yield table print file (YIELDTBL.FIL) contains information on the yield tables and on any errors the FORPLAN program finds in the yield data (YIELD.CAC). If any errors are detected in either file, they are flagged with the characters *E* in the first three columns of the corresponding print file. Check columns 1-3 for *E*. Following this indicator appears an error message number and brief explanation that can be cross checked with Appendix H in the User's Guide (LMP 1992). Warning messages are flagged in a similar fashion with *W* in the first three columns. In addition, the User's Guide has detailed information concerning each error message and warning located in the chapter corresponding to the data section in error.

Initially, you may want to locate the "error check points" found in MAT.FIL to determine the number of errors found in certain sections of the data. In all cases, a message listing the total number of errors detected in both data and yield files are printed at the end of this file, and this is denoted as the "FINAL CHECK POINT."

NOTE: Errors in FORPLAN data or yields tend to "cascade," that is, an error in one part of the data set often leads to FORPLAN "finding" errors in later parts of the data set. These later errors are often not real but occur because the programs do not have all of the information they need because of the original error. Consequently, one should always start by fixing the first errors listed; often fixing a few errors up front will eliminate many (sometimes hundreds) of later errors without any additional changes in the data or yield files. FORPLAN provides you with the capability to control error proliferation by terminating an execution if more than a specified number of errors are encountered. Refer to the IFSTOP input section of the user's guide (LMP 1992) for more information on this issue.

Correct any errors (subject to the note above) and rerun your batch file. Once the data set and yield file are error free, you may generate your matrix.

B. GENERATING THE MATRIX

Once a data set and yield file are error free, a matrix can be generated. This task can require a considerable amount of machine resources for large data sets, so be prepared to have your machine running for one to three hours during the matrix generation phase, and be sure that you have adequate file storage space on your hard drive.

Using BATGEN to build a batch file to generate a matrix for the CAC data set results in you being asked the same questions we described for a data edit run. With the exception of two questions, all answers will be the same. For the first of these (name of the batch file), you may wish to choose a different name, for example, ZONEGEN.BAT as we assume here. For the second (type of run), the correct answer is **2. Matrix Generation.**

The contents of ZONEGEN.BAT are straightforward, especially if you have read and understood the information given in Chapter 3 on COMPLETE.BAT. Note that both MATGENM1.EXE and MATGEN.EXE are run.

Once you have executed ZONEGEN, matrix generation for the CAC data set is complete. In addition to a log file (MAT.LOG) and two print files (MAT.FIL and YIELDTBL.FIL) such as those created during the data edit run, 9 matrix files (MATRX.*) or LP data files are created. We saw something similar to this operation in Chapter 3 with the SIX sample data set (fig. 5). Matrix file sizes in bytes for each data set are

FILE Name	*SIX Size*	*CAC Size*
MATRX.ROW (8)	91	1,289
MATRX.OTH (12)	9	3,505
MATRX.CAC (13)	0	34,258
MATRX.RX (9)	608	62,436
MATRX.RHS (10)	157	385
MATRX.RNG (11)	8	46
MATRX.BOT (22)	8	882
MATRX.BCA (23)	0	76
MATRX.BRX (24)	8	1,000

The files are listed in the order they are concatenated for the LP solvers (see Chapter 5) and the numbers in parentheses are the old file extension names used by Release 13, provided for cross reference purposes. All files are written in the Mathematical Programming System (MPS) format. Some contain MPS header cards as well as data because, unlike Release 13, Release 14.2 writes these out in the matrix files (see Chapter 2). Note that two of the files for data set SIX are empty and that four contain less than ten bytes of information (MPS header cards but no LP data). Three files (MATRX.ROW, MATRX.RX and MATRX.RHS) always contain LP data, whereas the other six files may or may not, depending on the options invoked in a given FORPLAN data set.

In general, the contents of these files are as follows:

MATRX.ROW	Contains the NAME and ROWS header cards, the row names of all rows in the model, and the type of relationship. (E=equal, L=less than, G=greater than, N=free row.)
MATRX.OTH	Contains the COLUMNS header card and column data for all columns in the model that do not represent prescriptions or coordinated allocation choices (CACs). Examples include transfer columns that represent periodic output totals and demand columns.
MATRX.CAC	Contains column data for all CAC implementation choices in the model. As with prescriptions, each timing choice for each CAC is represented by a separate column. This file is generated only for data sets with zones and can get quite large.
MATRX.RX	Contains the column data for prescriptions. All prescription timing choices that are generated are represented as separate columns in the model. This file details the intersection between each prescription timing choice and each row in the model. This file can get quite large (over 1 million lines). It often contains 60% to 80% or more of the entire model's data.

MATRX.RHS	Contains the right-hand-side (RHS) header card and the RHS values of all constraints with nonzero RHS values. If the constraint being modeled has an upper and a lower limit, only the lower limit will be included in this file.
MATRX.RNG	Contains the RANGES header card and the difference between the lower and upper limits of constraints whose right-hand-side values have been ranged.
MATRX.BOT	Contains the BOUNDS header card and any bounds defined for transfer and accounting columns (i.e., those columns defined in MATRX.OTH).
MATRX.BCA	Contains any bounds defined for CAC columns (i.e., those defined in MATRX.CAC). This file will be generated only for data sets with zones, and only if one or more CAC columns are bounded.
MATRX.BRX	Contains any bounds defined for prescription timing choice columns (i.e., those defined in MATRX.RX) and the ENDATA card.

These files are written using an extended MPS format. The extensions include (1) the use of embedded blanks in the row and column names and (2) the use of an alphabet of 61 characters (see Appendix F of the user's guide (LMP 1992) for the list of characters used). For more specifics on the MPS format, refer to Schrage (1989).

After you complete a matrix generation run, a good first step is to check the matrix print file (MAT.FIL) for errors at the final error check point by using a text editor. You may have had a successful data edit run checking for errors but additional errors may be found when the matrix is being generated. If you execute a matrix generation run and either program (MATGENM1.EXE or MATGEN.EXE) encounters data errors, the run will revert to an error checking run and the MATRX files will not be completed. If this happens, the partially written MATRX files should be deleted as part of your file management routine before beginning another matrix generation run. The distributed batch files will delete these partial files for you, as will those runstreams created by BATGEN.

CHAPTER 5. OBTAINING A SOLUTION

Once the LP matrix has been generated, it must be solved using an LP solution software package. Two commercially available packages, LINDO and C-WHIZ, are supported in BATGEN. Batch files for both of these solvers are included in the FORPLAN distribution package. Batch files using either solver can be generated with BATGEN. In this chapter we will discuss the use of these systems in conjunction with BATGEN and FORPLAN. We do not describe all the features and options of LINDO and C-WHIZ — this material is intended to supplement their documentation, not to replace it. Before we begin the discussion, we present an overview of LP solver testing results that led to the selection of these packages.

A. LP SOLVER TESTING

Scientists at the Rocky Mountain Station have tested the capabilities of three commercially available LP solvers (LINDO, CPLEX and C-WHIZ) on microcomputers. They utilized a suite of 15 models (mostly FORPLAN) of widely different sizes and formulations. LINDO was selected for testing because when Release 13 was downloaded it was the only solver available to run on PC's that could handle FORPLAN-sized models. Hence, LINDO was the only solver supported by that package (Kent et al. 1992). CPLEX and C-WHIZ were chosen because they are both widely recognized as state-of-the-art LP solvers that were among the first (after LINDO) to market PC versions of their software that could solve large FORPLAN models.

Findings from the testing include

1. On almost all models, CPLEX and C-WHIZ solved significantly faster than LINDO (sometimes 200 to 500% faster). CPLEX and C-WHIZ load the entire problem into RAM while LINDO uses virtual paging to swap pieces of the problem in and out as needed. This swapping appears to be a major reason for LINDO's slower performance.

2. Because CPLEX and C-WHIZ load the entire problem into RAM, they require adequate RAM for problem loading or they will not run. LINDO requires less RAM for a given problem because of its page swapping.

3. C-WHIZ requires less RAM to load a given problem than does CPLEX (see results below for a modified Tongas National Forest model).

4. LINDO has fixed limits on the size of problem it can load (for example, 2,000,000 nonzero coefficients maximum) while CPLEX and C-WHIZ are limited only by the RAM available.

5. C-WHIZ and CPLEX offer advanced features that are not available or that do not function correctly in LINDO, such as advanced bases and a wider variety of sensitivity analysis techniques.

6. On larger models using default tolerance settings, C-WHIZ was usually faster than CPLEX (usually 50 to 100%). CPLEX tolerances usually could be fine tuned to make it perform almost as well as C-WHIZ, but such tuning was model specific.

A Compaq 386/25 was the base machine used to test LP software. This machine had both INTEL 80387 and Weitek coprocessors and tests were run with both. In general, models solved 25 to 75% faster on the Weitek than on the Intel 80387 with the greatest improvements tending to show up with the larger models. However, in more recent testing on 80486 machines the difference in performance between the two coprocessors is much smaller, usually amounting to less than a 10% speed gain for the Weitek.

There was close agreement in objective function values and constraint achievement levels across the various LP solvers and models (usually a 0.2% difference or less), but alternative optima resulted in different land allocations in most models. Overall, all three of these products demonstrate substantial LP model solving capability on MS DOS-based microcomputers. From these results, it appears that C-WHIZ is the package of choice, especially for larger models. For most smaller models, LINDO appears to be a very satisfactory alternative.

Tongass National Forest Model Test Results

A modified model from the Tongass National Forest was the largest test problem used in terms of nonzero coefficients. Following are some statistics from this model:

MODEL SIZE

ROWS	2,508
COLUMNS	40,161
# OF COEFFICIENTS	2,155,179

Performance results for this model using the three solvers were as follows:

PERFORMANCE RESULTS
(times are wall clock times in hrs., mins., and sec.)

	8 MB RAM	***16 MB RAM***
CPLEX on COMPAQ 386/25	Insufficient RAM to load problem	Insufficient RAM to load problem
386 LINDO	Problem too large for LINDO to load	Problem too large for LINDO to load
C-WHIZ on COMPAQ 386/25	Insufficient RAM to load problem	5:47:45
C-WHIZ on COMPAQ 486/33 (on-board INTEL coprocessor)	Insufficient RAM to load problem	1:35:42

The first three sets of results are from using the COMPAQ 386/25 with the INTEL 80387 math coprocessor. The last set of results were obtained using a COMPAQ 486/33, which was able to solve the model in about one-third of the time required for the 386/25. Note the results suggest that C-WHIZ is more efficient than CPLEX in its use of RAM to store and solve a problem, as C-WHIZ required 14.5 MB to deal with the Tongass model while CPLEX was not able to fit the model into 16 MB.

Region 6 Test Results

Region 6 analysts also evaluated LINDO and C-WHIZ. They found that the C-WHIZ software reduces the size of the matrix to be solved by screening out non-binding columns and rows (using a feature called PRESOLVE; see discussion later in this chapter), while the LINDO software does not. This process (often referred to as matrix analysis) also allows C-WHIZ to detect certain kinds of infeasibilities (known as patent infeasibilities) without starting the optimizer or solution phase. This procedure can save substantial computing time, particularly in comparison to LINDO, which can have difficulties in identifying infeasibilities in large models.

For example, Region 6 personnel attempted to solve the Siskiyou National Forest model consisting of more than 60,000 columns and 1,000,000 coefficients. After about eight hours of processing time and 100,000 iterations, LINDO was unable to solve the model or to identify it as being infeasible. C-WHIZ required less than two hours to determine that the problem was patently infeasible. C-WHIZ also flagged the infeasible rows, thus identifying the location of model adjustments needed to make it feasible. Once this was done, C-WHIZ solved the problem in 90 minutes.

B. USING LINDO

Users of Release 14.2 may use LINDO to solve their LP models, as it is supported by both the distributed batch files and by BATGEN (see Chapter 3). LINDO is available from

LINDO Systems, Inc.
P.O. Box 148231
Chicago, IL 60614
(312) 871-2524

Cost to agency users is $750.00 under a site license agreement. You should specify the 386 LINDO version, as there are older versions of the software that will not work with FORPLAN. LINDO is available in both 80387 (or 80486) (LINDO87.EXE) and Weitek (LINDOW.EXE) math coprocessor versions. LINDO will accept LP models up to the following size limits:

— 2,000,000 nonzero coefficients including those for slack variables
— 100,000 columns including slacks
— 8192 rows

As noted in Chapter 2, LINDO will not accept embedded blanks in the row and column names; consequently, batch files execute the LMPSFX.EXE utility to fill in embedded blanks with underscores.

When you purchase the LINDO software, it will come with two guides: a user's manual (Schrage 1989) and another titled *Linear, Integer, and Quadratic Programming with LINDO* (Schrage 1986). We suggest that you read the section in *Linear, Integer, and Quadratic Programming with LINDO* on LP, and become familiar with the LINDO user's manual. In the discussion that follows, we highlight some of the features of LINDO described in these guides as they apply to FORPLAN.

To install LINDO on your system, follow the instructions that come with the software. Remember, LINDO's directory of residence should be in your path statement. For example, if you load LINDO into C:\LINDO, then your AUTOEXEC.BAT file should contain the following as part of your path statement:

```
PATH C:\;C:\DOS;C:\FORPLAN;C:\LINDO
```

Continuing Our Example

In Chapter 4 we outlined the process for generating an LP matrix using the CAC sample data. Now we consider the use of BATGEN and LINDO to solve that model. In Chapter 3 we presented an example of the LINDO-related questions asked by BATGEN in the context of building a complete run batch file. The questions we discuss here are similar to those.

As in Chapter 4, we assume that BATGEN is in your path and that the CAC LP data (matrix chapters) are in

```
C:\FORPLAN\CAC
```

as suggested in Chapter 2. Also, we assume that you are executing BATGEN from that directory.

Upon executing BATGEN, you will first see the introductory screen shown in figure 1. As before, select option 1, Runstream Generator. The next question asks for the name of the batch file to be created:

> **Enter runstream filename (a ".BAT" extension is required and is assumed if not typed in)**
> **(or hit <CR> for default of RUN.BAT):** LP

This results in the creation of the file LP.BAT. The next screen contains the list of possible run types (fig. 2) and we select **5. LP** for LP model solution. BATGEN then asks

> **LP SOFTWARE**
> **1. LINDO**
> **2. C-WHIZ**
> **Enter number of LP solver:** 1

We select LINDO.

The next question relates to the selection of objective functions:

> **OBJECTIVE FUNCTIONS**
> **RETRIEVED FROM MATRX.ROW**
> **1. OB1PNW20**
> **2. OB2TIM20**
> **Enter number of choice:** 1

This question differs from the objective function question asked in Chapter 3. In that case, no matrix had been generated (recall we were doing a complete run) and BATGEN scanned the FORPLAN input data to determine possible objective function names. Because the matrix has been generated BATGEN scans the row names in MATRX.ROW. We select the first objective function OB1PNW20.

The next question is

> **Do you want to:**
> **1. Maximize**
> **2. Minimize**
> **Enter number of choice:** 1

We have selected maximization. BATGEN then asks

> **Enter max iterations (or hit <CR> for default of 100000): 4000**
> **Do you want screen output? (enter a y or hit <CR> for "no"):**

We specify an iteration limit of 4,000 and select the default of no output from LINDO to the screen. The CAC model is small so that a limit of 4,000 iterations is more than sufficient. In most cases the default answer of 100,000 iterations should be sufficient even if your model is large. By choosing the default answer to the second question, all output from LINDO (i.e., matrix statistics, iteration log information, final iteration count) will be contained in the file LP.FIL as is discussed below. If you specify sending LINDO output to the screen, then the file LP.FIL will not be created.

Finally, BATGEN will ask if you wish to create another runstream in the batch file as we have discussed in Chapter 3. We answer the question "**NO**" in this case.

More on Objective Function Selection

We have discussed BATGEN's ability to specify possible objective functions by scanning the FORPLAN data set (Chapter 3) and by scanning the file MATRX.ROW. If that file does not exist and we have specified run type **5. LP**, then BATGEN will ask:

```
**** The MATRX.* files have to be available to the specified directory
**** before the LP can be run.
Do you want to:
1.  Specify a sub-directory and copy MATRX.* files
2.  Run MATGEN, then the LP solver
3.  Quit now and regroup
    Enter a 1, 2 or 3:  1
```

Here we have selected the first option so BATGEN then asks:

```
Enter sub-directory for existing MATRX.* files:  (Ex. D:\MATFILE, up to 32 characters)
C:\FORPLAN\OTHRDAT
```

Here we have specified the path to the desired LP data files. BATGEN then responds with

```
OBJECTIVE FUNCTIONS
RETRIEVED FROM C:\FORPLAN\OTHRDAT\MATRX ROW
1.  OB1PNW20
2.  OB2TIM20
    Enter number of choice:  1
```

Here we have selected the first one listed. Also, in this case, the .BAT file built by BATGEN will copy all MATRX files from C:\FORPLAN\OTHRDAT into the MPS.FIL (see below for a discussion of this file) before LINDO is executed.

If you select the second option given above (Run MATGEN) then you will go through the sequence of questions described in Chapter 4. Selecting the third option terminates BATGEN.

Our Example Continued - The Files LP.BAT and LPINP.DAT

Executing BATGEN as just described results in the creation of 2 new files, LP.BAT (fig. 6) and LPINP.DAT (fig. 7). The contents of LP.BAT are similar to the LP portion of COMPLETE.BAT as described in Chapter 3. In addition to numerous file housekeeping commands and the execution of the TIM.EXE (discussed in Chapter 3) and LINDO (actually named LINDO87.EXE), this file also contains brief descriptions of the pre- and post-processors for LINDO (LMPSFX.EXE and LSOLFX.EXE) and executes these programs as well. We will discuss these programs below.

LPINP.DAT contains information specific to the problem being solved as well as other control information read by LINDO. The first three lines in figure 7 set print control parameters for LINDO. The next line tells LINDO to read in the problem in MPS format from a file called MPS.FIL. The next two lines declare the objective function name and indicate that it is to be maximized. The next line tells LINDO to print the solution results into a file called FILE.SOL. The next two lines relate to the limit of 4,000 iterations we specified when we ran BATGEN. The next two lines are controls used by LINDO to specify formats and the solution information that is to be printed in FILE.SOL. The last line terminates the LINDO run. More details on LINDO input conventions can be found in the LINDO user's manual (Schrage 1989).

The LINDO Preprocessor LMPSFX.EXE

Referring to figure 6, note that LMPSFX.EXE is executed before LINDO. This program is necessary, in part, because the FORPLAN matrix generator writes the LP data in up to 9 separate files MATRX.*, and LINDO requires that the entire LP model be in a single file (Chapter 4). Also, LINDO is unable to accept row and column names that contain embedded blanks. Thus, LMPSFX.EXE accomplishes two tasks:

1) replaces all embedded blanks with underscores, and
2) concatenates the MATRX.* files into a single file, MPS.FIL, which is read by LINDO.

```
Rem    LP SOLVER (LINDO, 80387 VERSION)
del lp.fil
del mps.fil
del file.sol
del sol.fil
del lp.log
echo off
Rem    The following preprocessor for LINDO is required instead of a COPY
Rem    command due to Lindos inability to handle imbedded blanks in the
Rem    row & column names. This utility replaces MPSLND.EXE which is not
Rem    compatible with FORPLAN Release 14.1.
echo on
tim.exe >lp.log
lmpsfx.exe
echo off
Rem    Execute LINDO and the postprocessor LSOLFX.EXE to prepare a SOL.FIL.
Rem    Note that this batch file requires that problem specific info be in a
Rem    file named LPINP.DAT (analogous to the MPS.CR file in C-WHIZ). This
Rem    batch file also assumes that the LPINP.DAT file has identified the
Rem    input matrix file as MPS.FIL and the intermediate LINDO solution file as
Rem    FILE.SOL. FILE.SOL is read by LSOLFX.EXE which is the postprocessor
Rem    that produces SOL.FIL for FORPLAN.
echo on
tim.exe >>lp.log
lindo87 <lpinp.dat  >lp.fil
lsolfx.exe
tim.exe >>lp.log
del mps.fil
del file.sol
```

Figure 6.—Listing of LP.BAT for executing LINDO.

```
PAGE 0
WIDTH 132
TERSE
RMPS MPS.FIL
OB1PNW20
MAX
DIVERT FILE.SOL
GO 2000
   2000
RPRI N T P R L U D :N = '%%%%%%%%'
CPRI N T P R L U D :N = '%%%%%%%%' .AND. P > 0
QUIT
```

Figure 7.—Listing of LPINP.DAT.

Note that the next to last command in LP.BAT (fig. 6) deletes MPS.FIL. This task is performed because during the time MPS.FIL exists, your LP model is stored twice, once in the MATRX.* files and again in MPS.FIL. For example, if you have a large model requiring 50 MB of disk space, 100 MB of free hard drive space will be required to store the two copies of your model while MPS.FIL exists. Unfortunately this problem also exists with C-WHIZ.

Running LP.BAT

When you execute LP.BAT, three new files are created, LP.LOG, LP.FIL and SOL.FIL. We noted the same 3 files in Chapter 3 when we worked with the SIX data. File sizes for each data set are

FILE Name	*SIX Size*	*CAC Size*
SOL.FIL	1,085	16,049
LP.LOG	81	81
LP.FIL	548	726

As noted in Chapter 3, the LP.LOG file contains data on the date and execution time for LINDO to run. For example, on our COMPAQ 486/33, LINDO requires approximately 19 seconds to solve the CAC data set. If your model is large, it may take many hours to solve using LINDO. In some cases, LINDO has been unable to solve the problem at all (see Region 6 test results discussed at the beginning of this chapter).

NOTE: Any previous solution file (SOL.FIL) you have in this directory will be deleted. If you wish to keep a previous solution file move it to another directory or change its name.

The LP.FIL File

This file (fig. 8) contains summary statistics for the problem you have asked LINDO to solve (in this case, the CAC model), as well as limited solution result information. The first eight lines in the figure contain no useful information. They are responses LINDO would normally write to the screen as it determines which row to use as an objective function. Line 9 indicates that the CAC model has 89 rows and 640 columns including slacks.

The next line indicates the number of nonzero elements (2,117) and the nonzero constraint element count (1,527). Both numbers include slack variable coefficients and the latter excludes objective function coefficients. The density measure is the proportion (not the percent) of the possible number of elements in the problem that are nonzero. The next line contains the absolute value of the largest and smallest elements in the matrix. LINDO does not perform any scaling of the LP matrix, nor does it perform any matrix analysis or reduction. You should scale rows and columns by careful selection of units in which the outputs are expressed to avoid numerical problems. For example, if you are interested in tracking salmon smolts, it might be best to use thousands of smolts as the unit in which yields are defined, rather than smolts. A rule of thumb is that there should be no nonzero coefficient whose absolute value is greater than 100,000 or less than 0.0001. If the absolute value of the difference between the largest and smallest values is greater than 100,000, LINDO will automatically print a warning (Schrage 1989). Note that the warning is printed in figure 8. The problem usually will solve correctly even in the presence of the warning message.

NOTE: LP solvers work most efficiently if the number of unique values for coefficients in the matrix is small. One way to ensure this requirement is to combine the use of significant digits with your knowledge of the precision of the yield and cost information that goes into FORPLAN. For example, FORPLAN data sets have been observed where economic data has been carried out to the nearest 1/1000 of a cent (e.g., an output with a return of $12.63247 per so many units). Rounding such a number to the nearest dollar (or even the nearest 10 dollars) often would make more sense and be more consistent with your actual knowledge.

```
: : : : NAME

CANDIDATE OBJECTIVE ROW(S) IS(ARE):
OB2TIM20
OB1PNW20
LCSE2
LCSE1
WHICH IS IT?
? MAX OR MIN ?
? ROWS=             89 VARS=             640 NO. INTEGER VARS=          0
NONZEROES= 2117 CONSTRAINT NONZ=   1527(  684 ARE +- 1)  DENSITY=0.037
SMALLEST AND LARGEST ELEMENTS IN ABSOLUTE VALUE= 0.143090E-04  0.784000E+07
NO. < :             0 NO. =:        88 NO. > :      0, OBJ=MAX, GUBS <=       21
SINGLE COLS=        24
    WARNING:  PROBLEM IS POORLY SCALED. THE UNITS
OF THE ROWS AND VARIABLES SHOULD BE CHANGED SO
THE COEFFICIENTS COVER A MUCH SMALLER RANGE.
: : LP OPTIMUM FOUND AT STEP          93
OBJECTIVE VALUE = -15990.7490
: 2000
: : :
```

Figure 8.—LP.FIL produced by LINDO for the CAC data set.

The next line shows the count of the number of rows of each type: less than, equal, and greater than. It also indicates that the objective function is to be maximized. The GUB statistic is a measure of problem simplicity. A problem for which the GUB statistic is high relative to the total number of rows tends to be easier to solve, all other factors being equal. The next line contains the "single cols" statistic that is a count of the number of variables that appear in only one row. These variables are usually slack variables.

The first line after the warning mentioned above indicates the status of the solution (feasible, optimal, infeasible, etc.), and at which step (iteration) the optimum, if any, was found. If the solution status is optimal, the problem has been solved successfully, and the SOL.FIL should be ready to use to generate reports. If the status is infeasible, then further investigation will be necessary. The next line contains the optimal objective function value. The last two lines contain no meaningful information.

The LINDO Post Processor LSOLFX.EXE

In our discussion of LP.BAT (fig. 7) we noted that the detailed solution results are diverted to a file called FILE.SOL. LSOLFX.EXE reads the results contained in FILE.SOL and removes the underscores that were added by LMPSFX.EXE to the row and column names. LSOLFX.EXE also reformats the information in FILE.SOL so that it can be read by the FORPLAN report writer. This utility works somewhat differently from its predecessor used with Release 13 (Kent et al. 1992). See Chapter 2 for more details on this.

The SOL.FIL File

This file contains the optimal objective function value and information associated with each row and column in solution. It is formatted so that the FORPLAN report writer can interpret it. Figure 9 contains portions of the SOL.FIL produced by LSOLFX.EXE for the CAC data set. The four sets of three vertical dots in the figure denote deleted portions of the file performed for ease of presentation. The full SOL.FIL contains 141 lines.

In figure 9, note that the three words OPTIMAL, ENDROW, and ENDALL appear to the left of all other information. The first denotes the status of the solution (other possibilities include UNBOUNDED, UNFINISHED, or INFEASIBLE) and the number to the right (-15590.7490) is the optimal objective function value. ENDROW denotes the end of information on rows and ENDALL denotes the end of the file.

The information in SOL.FIL between the objective function/solution status line and the ENDROW line relates to rows, with one line for each row in the problem. Thus, the complete SOL.FIL contains 89 lines in this section because, as we noted in our discussion of LP.FIL above, there are 89 rows in the CAC model. The information after the ENDROW line applies to all LP columns (decision variables) that are in solution at a value greater than zero. As was noted in figure 7, there are 640 LP columns in the CAC model. However, only 49 of those have positive values in the optimal solution, so there are 49 lines of LP column information in the complete SOL.FIL.

The first column of information contains the row or LP column name. For example, AA037 refers to the acreage row for analysis area 037. Refer to Appendix F in the FORPLAN user's guide (LMP 1992) for detailed information on interpretation of row and LP column naming conventions.

The next column contains the row or LP column status. For example, an FR refers to a free row that is unconstrained. Notice that only the objective function row is free or unconstrained because LINDO does not retain any other free rows that may be in the matrix. Refer to the LINDO user's guide (Schrage 1989) for a complete interpretation of status codes.

The next column contains the "activity" value. This number is the actual value of each row and LP column in solution. The value associated with a constraint can be compared with the RHS value to determine how close the constraint is to being binding. The value of a prescription column will tell you the number of acres allocated to it. In figure 9 the value 2400.00 associated with row AA037 signifies that 2400 acres have been allocated to prescription columns in analysis area 037. The value 1080.00 for the column 1037TF 1 signifies that 1,080 acres (of the 2,400) have been allocated to that prescription column.

The next SOL.FIL column contains the "slack" value for each row and the objective function coefficient for each LP column. The slack values will tell you how far away from an upper or lower bound the row or column activity level is. For example, if a constraint row has an upper bound of 300 units and an activity of 126 units, the slack value is 174 units. In other words, the row value could have increased 174 units before it was constrained by the upper bound.

The interpretation of this information is somewhat different for LP columns. In our example, prescription column 1037TF 1 has an objective function coefficient of 0.39033. This value means that each additional acre allocated to this prescription column would increase the objective function value by 0.39033 units.

The next two columns contain the upper and lower limits on rows and columns. If a row is an equality row (as are the analysis area acreage rows) the limits are equal (and in the case of an analysis area acreage row are equal to the analysis area acreage). In the case of the objective function, the asterisks in the lower and upper bound columns imply no lower or upper bound. For LP columns, the lower limit of 0.0 is a reflection of the non- negativity or logical constraints usually present in an LP. The asterisks imply no upper bound. If an LP column has lower (other than zero) or upper bounds defined, these bounds will be reflected here.

The last SOL.FIL column contains the shadow prices for rows. For a given row, this value is the rate at which the objective function will increase as the right-hand-side value of that constraint is relaxed one unit and assuming that the optimal solution does not change. This value is usually expressed as the change in the objective function value for a one-unit change in the right-hand-side value. Notice that analysis area 037 has a shadow price of 0.54481. This value means that the objective function would increase 0.54481 units if another acre of analysis area 037 was made available.

NOTE: The FORPLAN report writer will read all information in the SOL.FIL and print information pertaining to rows and "other columns" (those in the file MATRX.OTH, see Chapter

```
OPTIMAL  -15990.7490
          OB1PNW20  FR    -15990.75000  15990.75000  ***************************************      -1.00000
          LCNON     LN  7840000.00000       .00000  7840000.00000  7840000.00000         -.00024
          LCTOG     LN         .00000       .00000         .00000         .00000          .00003
          DEDO3  1R LN         .00000       .00000         .00000         .00000         -.02466
          DEDO3  2R LN         .00000       .00000         .00000         .00000         -.01666
  .
  .
  .
          DEDO5  3R LN         .00000       .00000         .00000         .00000         -.00546
          PC037    1 LN     1080.00000       .00000     1080.00000     1080.00000        -.15448
          AA037     LN     2400.00000       .00000     2400.00000     2400.00000          .54481
          AA052     LN     1000.00000       .00000     1000.00000     1000.00000          .35648
          AA883     LN     4500.00000       .00000     4500.00000     4500.00000          .04970
          AAS.0.    LN        1.00000       .00000        1.00000        1.00000    -22865.00000
          R012  MN1 LN         .00000       .00000         .00000         .00000         -.00126
          R012   RD1 LN         .00000       .00000         .00000         .00000         -.00101
  .
  .
  .
          R092  MN4 LN         .00000       .00000         .00000         .00000         -.00126
          R092   RD1 LN         .00000       .00000         .00000         .00000         -.00101
          AZ       20 LN     2640.00000       .00000     2640.00000     2640.00000          .76989
          AZ       80 LN     2800.00000       .00000     2800.00000     2800.00000          .78447
          AZ       90 LN     2640.00000       .00000     2640.00000     2640.00000          .72636
          AZ      100 LN     2800.00000       .00000     2800.00000     2800.00000          .87129
ENDROW
          DEDO3  11 LN    18000.01000       .02466         .00000   200000.00000          .00000
          DEDO3  21 LN    20100.01000       .01666         .00000   220000.00000          .00000
          DEDO3  31 LN    20100.01000       .01125         .00000   275000.00000          .00000
  .
  .
  .
          DEDO5  21 LN    16800.03000       .05019         .00000   135000.00000          .00000
          DEDO5  31 LN    16800.03000       .00546         .00000   147000.00000          .00000
          A       23 1 LN     2640.00000     -.31400         .00000*********************          .00000
          A       84 1 LN     2800.00000     -.31496         .00000*********************          .00000
          A       92 1 LN     2640.00000     -.02238     2640.00000     2640.00000          .57344
          A      102 1 LN     2800.00000     -.02936         .00000*********************          .00000
          1037TF    1 LN     1080.00000      .39033         .00000*********************          .00000
          1037P1    1 LN     1320.00000      .54481         .00000*********************          .00000
  .
  .
  .
          M083GB21  LN      569.99600       .04970         .00000*********************          .00000
          M092MN  1 LN      180.01200      -.00126         .00000*********************          .00000
          M092RD  1 LN      180.01200      -.00101         .00000*********************          .00000
ENDALL
```

Figure 9.—Portions of the SOL.FIL produced by LINDO for the CAC data set.

4) under column headers that make the information easier to interpret than it is in SOL.FIL where there are no headers.

No Feasible Solution

If two or more constraints cannot be simultaneously satisfied, there is no feasible solution to the model in question. For example, $X1 < 3$ and $X1 > 10$ would result in no feasible solution. In general, infeasibilities indicate that conflicting demands, limits, or production capabilities exist in the model or data set, and that the problem cannot be solved as formulated. Resolution of an infeasibility requires determining the constraint or constraints that are the source of the problem. When appropriate, relaxing the offending constraint(s) once they are identified by sufficient amounts will resolve the problem.

If you have no feasible solution, you will see a message similar to the following in LP.FIL:

```
NO FEASIBLE SOLUTION AT SET 14
SUM OF INFEASIBILITIES = 8450.0
```

This message tells you that LINDO decided your model was infeasible at the 14th iteration, and that the sum of all the infeasibilities is 8,450 units. In this example there is only one infeasible row, and it has missed its lower bound by 8,450 units. Next, check the SOL.FIL file and look at the slack column (the fourth column). Referring to the discussion of a SOL.FIL given above, find the row with a slack of -8,450; this row is the offending or infeasible row. LINDO tried to solve the model, but could not find the last 8,450 units it needed to satisfy this constraint. Now look at the row name and decipher what this corresponds to in your FORPLAN data. Remember that the row- and column-naming conventions are in Appendix F of the FORPLAN user's guide (LMP 1992).

Analysis of infeasibilities is often very difficult, especially if more than one row or column is flagged as infeasible. Each case is different and there is no single approach. The best way to minimize the impact of sleuthing out an infeasible model is to carefully keep a log of the changes made for each run, recognizing that something to do with the changes made since the last feasible run has caused the infeasibilities. This task is easier when you make relatively few changes between each run. As a final note, LINDO, unlike C-WHIZ, is unable to detect patent or structural infeasibilities (see next section of this chapter for details).

Efficiency Considerations

The major factors contributing to increased solution time in solving an LP are 1) number of rows, 2) number of columns, 3) number of nonzero coefficients, 4) number of unique values for coefficients, and 5) poor matrix scaling. As discussed above, the number of rows, nonzero elements, matrix density, and scaling information can be found in LP.FIL.

Rows

All rows except free rows are of concern. In general, efforts to keep the row count to a minimum will help reduce solution time. The use of intervals, where appropriate for constraints in the later periods, is suggested. However, the implications of using intervals for the optimal solution should be tested.

Coefficients

The larger the number of nonzero coefficients or unique values of coefficients, the longer the solution time. Certain constraints, such as those created to model forestwide demand, often add a significant number of coefficients to a model. One example is known where four forestwide demand curves (defined for only the first five decades) nearly doubled the number of nonzero coefficients from 1.1 to 2.2 million.

Scaling and Matrix Reduction

LINDO does not do any scaling of the LP matrix; nor does it "reduce" the size of the matrix by detecting and removing nonbinding and redundant rows and columns. You must scale rows and columns to avoid numerical problems. If the matrix is poorly scaled, LINDO will almost always print the warning shown in figure 8. Poorly scaled models may be difficult to solve because of numerical precision problems.

Maximum Iterations

When you run BATGEN, you will be asked to specify the maximum number of iterations for LINDO. If you do not specify enough iterations, LINDO will terminate prematurely, and the last line in LP.FIL will read

WARNING, SOLUTION MAY BE NONOPTIMAL/NONFEASIBLE

If this termination occurs, you must rerun LINDO from scratch and specify a larger number of iterations. Note that you can raise the limit by editing the file LPINP.DAT.

Advanced Basis

Advanced basis runs do not work well with LINDO. LINDO does not actually save a complete basis. The LINDO user's guide (Schrage 1989) does discuss ways to execute advanced basis runs, but testing has shown that using these options requires more time than does solving the new LP from scratch. Consequently, BATGEN does not support this capability when using LINDO.

C. USING C-WHIZ VERSION 1.4

Users of Release 14.2 may use C-WHIZ Version 1.4 to solve their LP models. C-WHIZ is a product of Ketron Corp. and is based on their mainframe package MPS III and its WHIZARD optimizer. The package is available free of charge to agency users under a site license agreement. For information about obtaining a copy, contact the FORPLAN hotline:

(303) 498-1833.

Non-agency users may purchase the software by contacting Ketron Corp. at

Ketron Management Science Corp.
1700 North Moore Street, Suite 1710
Arlington, VA 22209
(703) 558-8701

C-WHIZ is currently available for use on personal computers and selected UNIX workstations. The microcomputer version of C-WHIZ requires an i386 or i486 CPU with at least 2 MB RAM, an Intel or Weitek math coprocessor, and DOS 3.1 or later. There is a problem size limit of 32,000 rows (no limit on the number of columns). Because no FORPLAN model to date has approached 32,000 rows, the real limit on problem size using C-WHIZ is the amount of RAM installed on your machine, because C-WHIZ loads the entire problem in RAM prior to solving it. While a number of factors influence the amount of RAM required for a given model, the most important one is the number of coefficients in the model. Below we discuss how you can determine the amount of RAM C-WHIZ needs to solve a problem.

When you acquire the C-WHIZ software, it comes with a manual (Ketron 1992) entitled *C-WHIZ-Linear Programming Optimizer.* We recommend that you read this manual to obtain a broad understanding of the C-WHIZ package and its capabilities. In our discussion here, we will focus on those aspects of C-WHIZ that relate directly to FORPLAN.

To install C-WHIZ, follow the instructions provided with the software. They guide you through using the INSTALL program and ensuring that C-WHIZ is in your path statement. The BATGEN program assumes that you have followed these directions.

Continuing Our Example

In Chapter 4 we outlined the process for generating an LP matrix using the CAC sample data. Now we consider the use of BATGEN and C-WHIZ to solve that model. Some of the questions asked by BATGEN are similar to those pertaining to LINDO that we discussed in Chapter 3 when we used BATGEN to build a complete run batch file.

As in Chapter 4, we assume that BATGEN is in your path and that the CAC LP data (matrix chapters) are in

C:\FORPLAN\CAC

Also, we assume that you are executing BATGEN from that directory. Upon executing BATGEN, you will first see the introductory screen shown in figure 1. As before, select option 1, Runstream Generator. The next question asks for the name of the batch file to be created:

Enter runstream filename (a ".BAT" extension is required and is assumed if not typed in)
(or hit <CR> for default of RUN.BAT): LP

Note that our answer results in the creation of the file LP.BAT. The next screen contains the list of possible run types (fig. 2) and we select **5. LP** for LP model solution. BATGEN then asks

LP SOFTWARE
1. LINDO
2. C-WHIZ
Enter number of LP solver: 2

We select C-WHIZ. The next question is

Do you want to do a ROLLOVER run?
(Hit <CR> for default of no):

Here we select the default of no. Rollover runs are discussed in detail below.

The next question relates to the selection of objective functions:

OBJECTIVE FUNCTIONS
RETRIEVED FROM MATRX.ROW
1. OB1PNW20
2. OB2TIM20
Enter number of choice: 1

This question differs from the objective function question asked in Chapter 3. In that case, no matrix was generated (recall we were doing a complete run) and BATGEN scanned the FORPLAN input data to determine possible objective function names. Here, because the matrix has been generated, BATGEN scans the row names in MATRX.ROW. We select the first objective function OB1PNW20.

The next question is

Do you want to:
1. Maximize
2. Minimize
Enter number of choice: 1

We have selected maximization.

BATGEN then asks

Do you wish to use an advanced basis as a starting point for this run? (y,n)
(Hit <CR> for default of no):

This question relates to the use of an advanced basis start and we select the default of no. We will discuss advanced basis runs below. The next question is

Do you wish to save the optimal basis? (y,n)
(Hit <CR> for default of no):

Again, this operation relates to the use of an advanced basis and will be discussed below.

Finally, BATGEN will ask if you wish to create another runstream in the batch file as we have discussed in Chapter 3. We answer the question "**NO**" in this case.

The File LP.BAT

Executing BATGEN as just described results in the creation of a new file, LP.BAT (fig. 10). When executed, LP.BAT begins by deleting several files to avoid potential file conflicts. Note that one of these files is SOL.FIL. If you have an optimal solution from a previous run that you wish to save, rename or copy it elsewhere before executing your batch file. The next few lines of LP.BAT are remarks that address the inclusion of certain MPS header cards in the file MPS.FIL (see Chapters 2 and 4). After the execution of TIM.EXE (discussed in Chapter 3) the DOS concatenated copy command is used to copy all of the LP matrix chapters produced by the FORPLAN matrix generator (see Chapter 4) into MPS.FIL. This procedure is necessary because C-WHIZ requires that all the LP data be in a single input file. LP.BAT then checks for the existence of MPS.FIL for the purpose of branching around the rest of the run if that file is not present. Next, there are more remarks that describe several of the files associated with C-WHIZ. More details on these files are given later in this chapter.

C-WHIZ Command Line Settings

After executing TIM.EXE again, the next line in LP.BAT (fig. 10) is

```
whiz -SA -SFFILE -SP "-POOB1PNW20" -x MPS.FIL
```

This line executes C-WHIZ and the items after the word whiz are command line settings of parameters and arguments. Thus, for many C-WHIZ parameters and arguments there are two ways to change default values: 1) make changes in a copy of MASTER.CR (this file is distributed with the C-WHIZ package, see Ketron 1992 for details) named MPS.CR, or 2) specify desired arguments on the command line at execution time. Some versions of C-WHIZ had limited capability to accept command line settings so it was necessary to make most modifications in MPS.CR. Version 1.4 offers greatly increased capability to use the command line option and BATGEN takes full advantage of this procedure.

The settings used in our example have the following meanings:

```
Rem     LP SOLVER (C-WHIZ, 80387 VERSION)
del lp.fil
del mps.fil
del file.sol
del sol.fil
del lp.log
del actfile.act
echo off
Rem     The following concatenated COPY will include BOUNDS and RANGES cards in
Rem     the MPS.FIL even if no bounds or ranges are created by the matrix
Rem     generator.  This causes no problems to recent releases of C-WHIZ or
Rem     LINDO.  This COPY replaces the utility MPSLND.EXE.
echo on
time.exe  >lp.log
copy matrx.row+matrx.oth+matrx.cac+matrx.rx+matrx.rhs+matrx.rng+matrx.bot+matrx.bca+matrx.brx  mps.fil
if not exist mps. fil goto error1
echo off
Rem     Execute C-WHIZ and the postprocessor SOLB2A to prepare a SOL.FIL.
Rem     Note that C-WHIZ requires that problem specific info be in a file named
Rem     MPS.CR (analogous to the LPINP.DAT file we use with LINDO). Also note
Rem     that C-WHIZ produces two files; ACTFILE.ACT, and SYSPRINT
Rem     ACTFILE contains a binary representation of the problem which
Rem     saves C-WHIZ from going through the input phase if you want to solve the
Rem     problem again. This file can be sizable so this batch file deletes it.
Rem     SYSPRINT contains matrix statistics and iteration results. It can be
Rem     fairly sizable based on the options set in MPS.CR. This batch file
Rem     renames the file from SYSPRINT to LP.FIL.  C-WHIZ produces an
Rem     intermediate solution file (assumed to be named FILE.SOL in this batch
Rem     file). FILE.SOL is read by SOLB2A which is to be a postprocessor that
Rem     produces SOL.FIL for FORPLAN.  Use MPS.CR to set the name to FILE.SOL.
echo on
copy %mps%\master.cr mps.cr
tim.exe >>lp.log

whiz -SA -SFFILE -SP "-POOB1PNW20" -x  MPS.FIL

if not exist file.sol goto error1
solb2a file.sol sol.fil
tim.exe >>lp.log
ren sysprint lp.fil
del mps.fil
del mps.cr
del file.sol
del mps.cr#
del mps.gv#
del actfile.act

goto end1

:error1
echo off
echo ERROR in LP SOLVER
:end1
echo End of LP SOLVER Run #1
```

Figure 10.—Listing of LP.BAT for executing C-WHIZ.

-SA	This setting ensures that only decision variables that are in the optimal basis will be included in LP.FIL and the solution file (SOL.FIL). Unlike LINDO, C-WHIZ will include basic variables that have zero activity levels in these files.
-SFFILE	This setting specifies a file name of FILE for the binary solution file FILE.SOL (SOL is the hardwired extension) written by C-WHIZ.
-SP	This setting prints the solution in the file SYSPRINT.
"-POOB1PNW20"	This setting declares the name of the objective function.
-x	This setting tells C-WHIZ to maximize the objective function.
MPS.FIL	This setting tells C-WHIZ that the LP input data is in MPS.FIL.

For more details on these and other parameters that can be set on the command line, refer to Ketron (1992).

Preparing the SOL.FIL

Normally after C-WHIZ solves an LP it writes out solution results. When LP.BAT is run, these results will be written in two places: in binary form in the file FILE.SOL and in the print file LP.FIL. Unfortunately this information is not formatted correctly for the FORPLAN report writer in either place; consequently additional processing of the solution results is required. Ketron programmers have developed a C-WHIZ post-processor named SOLB2A to serve this purpose and it is included with the Forest Service C-WHIZ distribution package. Non-agency users should specify that they wish to receive this utility when they order C-WHIZ.

In LP.BAT (fig. 10), the first thing that happens is to test for the existence of FILE.SOL to branch around SOL.FIL preparation if something went wrong and C-WHIZ did not terminate normally. Next, SOLB2A is executed. It reads the binary file, reformats the solution results, and writes the reformatted information out to SOL.FIL. The time log program TIM.EXE is run again and the file SYSPRINT contains the matrix statistics, iteration log, and solution results from C-WHIZ. SYSPRINT is renamed LP.FIL to conform with Release 14.2 standard file naming conventions. This step is necessary because C-WHIZ always writes this output to the file SYSPRINT.

The next few lines do some housekeeping by deleting unneeded files. Note that the first file deleted is MPS.FIL. This step is performed because during the time MPS.FIL exists, two copies of your LP model exist, one in the MATRX.* files, and the other in MPS.FIL. If you have a large model requiring, for example, 50 MB to store, 100 MB of free hard drive space will be required to store the two copies. The other files deleted include two scratch files created by C-WHIZ, MPS.CR#, and MPS.GV#. The file ACTFILE.ACT contains a binary representation of your LP model. If you wish to solve the same problem again, C-WHIZ can read this file more rapidly than it can MPS.FIL because ACTFILE.ACT is considerably more compact. This file may also be useful if you wish to work with other Ketron programs. For more information, see Ketron (1992). Batch files generated by BATGEN are not designed to utilize this capability — ACTFILE.ACT is merely deleted at the end of the run. The last few lines in LP.BAT relate to the error branches described above.

Running LP.BAT

When you execute LP.BAT, three new files (LP.LOG, LP.FIL, and SOL.FIL) are created. File sizes in bytes for these are

SOL.FIL	24,388
LP.LOG	81
LP.FIL	17,063

The LP.LOG file contains data on the run date and the time required by C-WHIZ to read and solve the CAC model (see Chapter 3 for more details on LOG files). For example, on our COMPAQ 486/33, C-WHIZ requires approximately 16 seconds to solve this model. If your model is large, it may take hours for C-WHIZ to solve it.

The LP.FIL File

This file contains a variety of information including summary statistics for the model being solved, timing information on problem input and solution phases, matrix reduction information, abbreviated iteration log information, and complete solution results. Here we briefly discuss the contents of the LP.FIL produced by C-WHIZ for our example. For a more complete description of the contents of this file (or SYSPRINT, as C-WHIZ names it), refer to Chapter 4 of Ketron (1992).

Figure 11 shows the information on the CAC model that C-WHIZ determines as it reads in MPS.FIL. The matrix statistics include the number of rows (92), the number of columns (639), and the number of nonzero elements (2,622). It also includes the name of the objective function, the fact it is to be maximized, and the names of the right-hand-sides (RHS 1), the bounds (BND-1), and the ranges (RNG 1). Finally, note that C-WHIZ took 2.7 seconds to input the CAC model.

```
KETRON PROPRIETARY SYSTEM MPSIII/pc
        Wed Dec 16      15:10:25     1992
1

    C-WHIZ V1.4 (Sep 04 1992): 0.00 SEC.

        MPS FORMAT MATRIX INPUT
          From Deck      ********
              On File     MPS.FIL

        To Problem ACTPROB
        On Actfile ACTFILE (ACTFILE.ACT)

        Rows        =    92
        Columns     =    639
        RHSs        =    1
        Ranges      =    1
        Bounds      =    1
        Elements    =    2622

    C-WHIZ INPUT:  2.97 SEC.

        OBJECTIVE      OB1PNW20 (MAX)
        RHS            RHS  1
        BOUND          BND -1
        RANGE          RNG  1
```

Figure 11.—C-WHIZ problem input section from LP.FIL.

Figure 12 contains information on storage requirements and results of the PRESOLVE process for the CAC model. Important items to note here are the BYTES IN INDEX ARRAY (11,774) and BYTES IN FREE CORE CHAIN (12,226,000). The first item is the number of bytes of RAM required to store the complete CAC model, while the second item is an approximate measure of the free RAM available to C-WHIZ as it solves the model. C-WHIZ conducts a PRESOLVE analysis of every problem to identify opportunities to reduce model size by eliminating redundant or unneeded constraints and decision variables. This analysis results in a smaller model that will solve more quickly. In some cases, PRESOLVE can identify an infeasible problem without having C-WHIZ try to solve it (see the section on infeasibilities below for more information). As figure 12 shows, the PRESOLVE process took 3.3 seconds for the CAC model and resulted in (among other things) the removal of 6 degenerate rows and 71 degenerate columns. For more information on the PRESOLVE process, see Chapter 6 of Ketron (1992). Finally, the NUMBER OF VALUES entry in figure 12 (732) indicates that there are 732 different nonzero values for coefficients in the CAC model, not counting +1 and -1.

```
NUMBER OF ELEMENTS                   2622
NUMBER OF VALUES                      732
BYTES IN INDEX ARRAY                11732
BYTES IN FREE CORE CHAIN         11965488
LONGEST IN COLUMN LENGTH               42
LONGEST RHS LENGTH                     10

                        TOTAL      NORMAL         .FREE.        FIXED     BOUNDED
ROWS                       92           0              4           87           1
COLUMNS                   639         614              0            2          23

     787   NON-UNIT COEFFICIENTS (excluding free rows and fixed columns)
           MIN COEF            0.00739          MAX COEF            2800.00000
           AVG COEF           22.15477          VARIANCE           58176.33939

PRESOLVE: 4.62 SEC.
       1   COLUMNS TRANSLATED BY SINGLETON ROWS.
       6   DEGENERATE ROWS INTERSECTING    71  DEGENERATE COLUMNS ARE IGNORED.
       4   COLUMNS TRANSLATED
      19   TRANSFER COLUMNS FREED.
     543   ACTIVE VECTORS IN THE MODEL.
```

Figure 12.—C-WHIZ matrix reduction (PRESOLVE) analysis results from LP.FIL.

NOTE: LP solvers work most efficiently if the number of unique values for coefficients in the matrix is small. One way to ensure this situation is to recall your old training in the use of significant digits and combine that with your knowledge of the precision of the yield and cost information that goes into FORPLAN. For example, FORPLAN data sets have been observed where economic data has been carried out to the nearest 1/1000 of a cent (i.e., an output with a return of $12.63247 per so may units). Rounding such a number to the nearest dollar (or even the nearest 10 dollars) often makes more sense and is more consistent with our actual knowledge.

Figure 13 shows the iteration log produced by C-WHIZ while solving the CAC model. A feasible solution was encountered at iteration 31 and the optimal solution at iteration 45. C-WHIZ then restores any constraints or decision variables removed by PRESOLVE. It then recalculates the optimal solution. The restoration and recalculation is part of the POSTSOLVE process and its purpose is to calculate the optimal solution and shadow prices for the original full problem as they may differ slightly from those obtained for the reduced problem created by PRESOLVE. The iteration log shown in figure 13 shows the minimal amount of information produced by C-WHIZ. On larger problems, more information is provided (see Chapter 4 of Ketron 1992 for more details).

```
ENTER FAST PRIMAL:   4.89 SEC.
    0   ITERATIONS    XFUNCT= 18833.19339        XNIF=          24 XSIF= 5855949.9200
PI-WEIGHTING DISCONTINUED
FEASIBLE SOLUTION:   5.00 SEC.
   31   ITERATIONS    XFUNCT= 17950.61085-
OPTIMAL SOLUTION:    5.11 SEC.
   45   ITERATIONS    XFUNCT= 15990.74860-
RESTORE ORIGINAL CONSTRAINTS
ENTER FAST PRIMAL:   5.17 SEC.
   45   ITERATIONS    XFUNCT= 15990.74860-
OPTIMAL SOLUTION:    5.17 SEC.
   45   ITERATIONS    XFUNCT= 15990.74860-
    4   DEGENERATE COLUMNS INSERTED
ENTER FAST PRIMAL:   5.17 SEC.
   45   ITERATIONS    XFUNCT= 15990.74860-
OPTIMAL SOLUTION:    5.22 SEC.
   45   ITERATIONS    XFUNCT= 15990.74860-
```

Figure 13.—C-WHIZ iteration log from LP.FIL.

Portions of the solution results for the CAC model are shown in figure 14. This information is similar to that contained in SOL.FIL (see discussion below), but is formatted somewhat differently. It is divided into two sections, the first section describing rows, and the second section describing columns. A complete description of the solution report column definitions is in Chapter 4 of Ketron (1992).

Also shown in figure 14 are C-WHIZ messages stating that the solution has been written in binary format to FILE.SOL. Additional information is given on RAM usage during the optimization just completed, and the time that optimization took in seconds. As noted above, FILE.SOL is the file SOLB2A reads to prepare SOL.FIL (see discussion below). The possible reduction figure (11,673 k bytes) is the number of kilobytes in free core (fig. 12) that were not utilized in solving the CAC model. In other words, this value is the amount of RAM installed on the machine but not needed to solve the CAC model (almost 12 MB). If you have a large model, you may want to review this number along with the two numbers shown in figure 12 to determine how near you are to running out of RAM.

The SOL.FIL File

This file contains the optimal objective function value and information associated with each row and column in solution, formatted so that the FORPLAN report writer can interpret it. Figure 15 contains portions of the SOL.FIL produced by SOLB2A for the CAC data set. The four sets of three vertical dots in the figure denote portions of the file deleted for ease of presentation. The full SOL.FIL contains 182 lines.

In figure 15 note that three words (OPTIMAL, ENDROW, and ENDALL) appear to the left of all other information. OPTIMAL denotes the status of the solution (other possibilities include UNBOUNDED, UNFINISHED, or INFEASIBLE) and the number to the right (-15,990.74902) is the optimal objective function value. ENDROW denotes the end of information on rows and ENDALL denotes the end of the file.

```
SECTION 1 - ROWS

NUMBER . . . ROW. .    AT   . . ACTIVITY.   . . . .SLACK  . . . LOWER      . . . UPPER   . . . .DUAL
                                              ACTIVITY.     LIMIT. . .     LIMIT. .    ACTIVITY.

   1   LCNON          EQ    7840000.00              .   7840000.00     7840000.00       0-0002
   2   LCTOG          EQ             .              .            .              .       0.0002 -
   3   LCSE1          BS             .              .         NONE           NONE            .
   4   LCSE2          BS             .              .         NONE           NONE            .
   5   0B1PNW20       BS   -15990.7486     15990.7486                        NONE         NONE
1.0000
   6   0B2TIM20       BS    92471.9549   92471.9549 -         NONE           NONE            .
   .
   .
   .
  89   AZ   20        EQ     2640.0000              .    2640.0000      2640.0000     0.7699 -
  90   AZ   80        EQ     2800.0000              .    2800.0000      2800.0000     0.7845 -
  91   AZ   90        EQ     2640.0000              .    2640.0000      2640.0000     0.6661 -
  92   AZ 100         EQ     2800.0000              .    2800.0000      2800.0000     0.8713 -

SECTION 2 - COLUMNS

NUMBER . COLUMN .      AT   . . ACTIVITY.      . . .INPUT  . . . LOWER      . . . UPPER  . .REDUCED
                                                COST. . .     LIMIT. . .     LIMIT. .     COST. . .

  93   DED03  11      BS    18000.0032         0.0247            .      200000.0                .
  95   DED03  21      BS    20100.0072         0.0167            .      220000.000              .
  97   DED03  31      BS    20100.0072         0.0113            .      275000.000              .
   .
   .
   .
 728   M083GB   1     BS      279.9984         0.0497            .           NONE               .
 729   M083GB 21      BS      569.9960         0.0497            .           NONE               .
 730   M092MN   1     BS      180.0120       0.0013 -            .           NONE               .
 731   M092RD   1     BS      180.0120       0.0013 -            .           NONE               .

BINARY SOLUTION WRITTEN TO FILE.SOL

POSSIBLE REDUCTION IN WORKING MEMORY:            11673 Kbytes

C-WHIZ END:               6.49 SEC.
```

Figure 14.—Portions of the C-WHIZ optimal solution from LP.FIL.

The information in SOL.FIL between the objective function/solution status line and the ENDROW line relates to rows, with one line for each row in the problem. The complete SOL.FIL contains 92 lines, one for each row in the CAC model. The information after the ENDROW line applies to all LP columns (decision variables) in solution (i.e., that are in the optimal basis). As was noted in figure 11, there are 639 columns in the CAC model. However, only 87 of those are in the optimal basis, hence there are 87 lines of LP column information in the complete SOL.FIL.

The first column of information contains the row or LP column name. For example, AA037 refers to the acreage row for analysis area 037. Refer to Appendix F in the FORPLAN user's guide (LMP 1992) for detailed information on interpretation of row and LP column naming conventions.

The second file column contains the row or LP column status. Possible status codes are (Ketron 1992)

BS	—	basic
LL	—	at lower limit

```
OPTIMAL  -15990.74902
           LCNON      EQ  7840000.00000       0.00000  7840000.00000  7840000.00000        0.00024
           LCTOG      EQ        0.00000       0.00000        0.00000        0.00000       -0.00022
           LCSE1      FR        0.00000       0.00000           NONE           NONE        0.00000
           LCSE2      FR        0.00000       0.00000           NONE           NONE        0.00000
           0B1PNW20   FR   -15990.74860   15990.74860           NONE           NONE        1.00000
           0B2TIM20   FR    92471.95491  -92471.95491           NONE           NONE        0.00000
.
.
.
           DED05  3R  EQ        0.00000       0.00000        0.00000        0.00000        0.00546
           PC037   1  LL     1080.00000       0.00000     1080.00000     1800.00000        0.15448
           AA037      EQ     2400.00000       0.00000     2400.00000     2400.00000       -0.54481
           AA052      EQ     1000.00000       0.00000     1000.00000     1000.00000       -0.35648
           AA883      EQ     4500.00000       0.00000     4500.00000     4500.00000       -0.04970
           AAS.0.     EQ        1.00000       0.00000        1.00000        1.00000    22865.00000
           R012  MN1  EQ        0.00000       0.00000        0.00000        0.00000        0.00126
           R012  RD1  EQ        0.00000       0.00000        0.00000        0.00000        0.00101
.
.
.
           R092  MN4  EQ        0.00000       0.00000        0.00000        0.00000        0.00126
           R092  RD1  EQ        0.00000       0.00000        0.00000        0.00000        0.00101
           AZ      20 EQ     2640.00000       0.00000     2640.00000     2640.00000       -0.76989
           AZ      80 EQ     2800.00000       0.00000     2800.00000     2800.00000       -0.78447
           AZ      90 EQ     2640.00000       0.00000     2640.00000     2640.00000       -0.66605
           AZ     100 EQ     2800.00000       0.00000     2800.00000     2800.00000       -0.87129
ENDROW
           DEDO3  11  BS    18000.00320       0.02466        0.00000   200000.00000        0.00000
           DEDO3  21  BS    20100.00720       0.01666        0.00000   220000.00000        0.00000
           DEDO3  31  BS    20100.00720       0.01125        0.00000   275000.00000        0.00000
.
.
.
           A      23 1 BS    2640.00000      -0.31400        0.00000           NONE        0.00000
           A      81 1 BS       0.00000       0.00000        0.00000           NONE        0.00000
           A      84 1 BS    2800.00000      -0.31496        0.00000           NONE        0.00000
           A      92 1 EQ    2640.00000      -0.02238     2640.00000     2640.00000       -0.51313
           A      93 1 BS       0.00000      -0.31400        0.00000           NONE        0.00000
           A     102 1 BS    2800.00000      -0.02936        0.00000           NONE        0.00000
           1037TF    1 BS    1080.00000       0.39033        0.00000           NONE        0.00000
           1037P1    1 BS    1320.00000       0.54481        0.00000           NONE        0.00000
.
.
.
           M083GB21   BS      569.99600       0.04970        0.00000           NONE        0.00000
           M092MN  1  BS      180.01200      -0.00126        0.00000           NONE        0.00000
           M092RD  1  BS      180.01200      -0.00101        0.00000           NONE        0.00000
ENDALL
```

Figure 15.—Portions of the SOL.FIL produced by SOLB2A for the CAC data set.

UL	—	at upper limit
EQ	—	fixed vector
FR	—	free column
**	—	infeasible

The third file column contains the "activity" value. This value is the actual value of each row and LP column in solution. The value associated with a constraint can be compared with the RHS value to determine how close the constraint is to being binding. The value of a prescription column will tell you how many acres have been allocated to it. In figure 15 the value 2,400.00 associated with row AA037 signifies that 2,400 acres have been allocated to prescription columns in analysis area 037. The value 1,080.00 for the column 1037TF 1 signifies that 1,080 acres (of the 2,400) have been allocated to that prescription column.

The fourth SOL.FIL column contains the "slack" value for each row and the objective function coefficient for each LP column. The slack values will tell you how far away from an upper or lower bound the row or column activity level is. For example, if a constraint row has an upper bound of 300 units and an activity of 126 units, the slack value is 174 units. In other words, the row value could have increased 174 units before it was constrained by the upper bound.

The interpretation of this information is somewhat different for LP columns. In our example, prescription column 1037TF 1 has an objective function coefficient of 0.39033. This means that each additional acre allocated to this prescription column would increase the objective function value 0.39033 units.

The fifth and sixth columns contain the lower and upper limits on rows and columns. If a row is an equality row (as are the analysis area acreage rows like AA037), the limits are equal (and in the case of analysis area acreage rows are equal to the analysis area acreage). For LP columns, the lower limit of 0.0 is a reflection of the non-negativity or logical constraints usually present in an LP. The word NONE implies no upper bound. If an LP column has lower (other than zero) or upper bounds defined, these will be reflected here.

The last SOL.FIL column contains the shadow prices for rows and reduced costs for LP columns. For a given row, this value is the rate at which the objective function will increase as the RHS value of that constraint is relaxed one unit, assuming that the optimal solution does not change. This value is usually expressed as the change in the objective function value for a one-unit change in the RHS value. Notice that analysis area 037 (row AA037) has a shadow price of -0.54481. Ignoring the minus sign, this value means that the objective function would increase 0.54481 units if another acre of analysis area 037 was made available. The presence of the minus sign is due to the fact that C-WHIZ actually minimizes the negative of the objective function when maximization is specified.

NOTE: The FORPLAN report writer will read all information in the SOL.FIL and print all information pertaining to rows and "other columns" (those in the file MATRX.OTH, see Chapter 4) under column headers that make the information easier to interpret than it is in SOL.FIL where there are no headers (see LMP 1992).

Advanced Basis Starts

C-WHIZ offers the capability to save and reuse an optimal basis as an advanced starting point for another run. This procedure can, especially in larger models, result in significant reduction in model solution time (refer to Ketron (1992) for more information on advanced bases). The BATGEN program supports this capability. To demonstrate this, we return to our example.

First, suppose we wish to save an optimal basis. To demonstrate how this task is done, we again run BATGEN to create a batch file that will execute C-WHIZ to solve the CAC LP model. This batch file will be identical to the one shown in fig. 10, except that it will also save the optimal basis once the model is solved. Execute BATGEN as described above, and answer all the initial questions as before until you come to the questions relating to advanced bases.

The first question is

Do you wish to use an advanced basis as a starting point for this run? (y,n)
(Hit <CR> for default of no):

As before, we answer "no," as we have yet to create a basis. The next question is

Do you wish to save the optimal basis? (y,n)
(Hit <CR> for default of no): Y

Here we answer "yes." The last question pertaining to advanced bases is

Enter filename (up to 8 characters).
Do not enter an extension, a ".BAS" extension will be added: OPTPNW

Our answer implies that the file containing the optimal basis is OPTPNW.BAS. C-WHIZ always assumes a .BAS extension for any file containing a basis.

The resulting batch file will look exactly like the file LP.BAT (fig. 10) except for the C-WHIZ command line:

```
whiz -SA -SFFILE -SP "-P00B1PNW20" -BPOPTPNW -x MPS.FIL
```

Note that this line is similar to the C-WHIZ command line shown in figure 10 except for the addition of the command line parameter -BPOPTPNW. The -BP denotes that an optimal basis is to be punched (written to a file in card images) and OPTPNW is the file name we gave BATGEN.

After the batch file has been executed, the file OPTPNW.BAS will reside in the same directory. Also, C-WHIZ will print a message to the effect that the basis was punched on OPTPNW.BAS. This message appears in LP.FIL immediately after the iteration log.

Next, we consider the case where we have an optimal basis from a prior run and wish to use it as the starting point for a new run. Specifically, we wish to use the optimal basis obtained when we maximized PNV for the CAC model in a new run where we maximize timber harvest. Again, using BATGEN we work our way through the questions until we encounter

Do you wish to use an advanced basis as a starting point for this run? (y,n)
(Hit <CR> for default of no): Y

Because we wish to use a prior basis, we answer yes. The next question asks for the file name:

Enter filename (up to 8 characters).
Do not enter an extension, a ".BAS" extension will be added: OPTPNW

Then BATGEN asks

Do you wish to save the optimal basis? (y,n)
(Hit <CR> for default of no): Y

Because we wish to save the new optimal basis, we answer yes. Finally, we are asked to provide a file name for this new basis:

Enter filename (up to 8 characters).
Do not enter an extension, a ".BAS" extension will be added: OPTTIM

This answer implies that the new basis will be stored in the file OPTTIM.BAS. While we do not show it here, you should keep in mind that we would specify the timber objective function (OB2TIM20) when BATGEN asks for an objective function.

NOTE: In the event that the prior basis file you name does not exist, BATGEN will ask

Basis file OPTPNW does not exist.
Do you want to:
1. **Re-enter advanced basis file name**
2. **Continue and do not use an advanced basis**
3. **Quit**

If you choose 1, you will again be asked to provide a file name; if you choose 2, BATGEN moves on to the next question pertaining to saving a new basis; and if you choose 3, BATGEN terminates execution.

When you run the batch file just created, C-WHIZ will read the prior basis contained in OPTPNW.BAS and once the problem is solved, write the new optimal basis out to OPTTIM.BAS. The read will be acknowledged with a basis inserted message that appears in LP.FIL in the problem statistics section. The write will be acknowledged in LP.FIL immediately after the iteration log. The batch file created for this run looks very similar to the one shown in figure 10 with only the C-WHIZ command line changing:

```
whiz -SA -SFFILE -SP "-POOB2TIM20" -BIOPTPNW -BPOPTTIM -x MPS.FIL
```

Here the -BIOPTPNW parameter denotes that a basis will be read in from OPTPNW.BAS and the -BPOPTTIM parameter denotes that a new basis will be written out to OPTTIM.BAS.

In our test, it took C-WHIZ 21 iterations to find the optimal solution for the CAC model with a maximize timber objective function using the PNV optimal solution as an advanced basis, as opposed to 27 iterations when solving the maximize timber problem from scratch. Overall, the time for the run increases slightly because of increased input and output associated with reading and writing to basis files and because of the fact that the optimization itself takes almost as much time (0.16 vs. 0.17 seconds) even when using the advanced basis. The CAC model is so small that most of the 16 or so total seconds spent in the LP run are spent performing I/O related tasks, so savings resulting from reducing solution time will be small. On large models where much more time is spent solving the problem, time savings resulting from the use of advanced bases can be substantial.

Rollover Runs

The term "rollover" is used to describe the process of making sequential LP runs to optimize more than one objective function using a single matrix. An example of the rollover process would be to maximize the flow of timber in the first period, and then maximize present net value over 20 periods, with the previously maximized timber amount fixed as a constraint. In general, the rollover process involves generating a matrix that includes all objective functions to be optimized, solving the problem using the first objective function, then incorporating the optimal objective function value as a constraint in the matrix, then optimizing the adjusted matrix given the second objective function and so on. Because of scaling and rounding within most LP solver programs, each objective function value should be reduced (if it will become a lower limit) or increased (if it will become an upper limit) by a small percentage before it is entered as an RHS value.

This process can be performed manually. Alternatively, BATGEN (in conjunction with C-WHIZ) has the ability to automate this process, using up to 5 objective functions. In addition, the FORPLAN distribution package (see Chapter 2) contains the executable program ROLLOVER.EXE, which is used to modify the LP matrix.

To demonstrate the rollover process, we continue with our example using BATGEN and the CAC data set. We wish to perform a rollover run where we maximize timber harvest for 20 periods and then maximize PNV, also for 20 periods, while holding the timber harvest level near the maximum level. Execute BATGEN and answer questions as though you are making a regular LP run using C-WHIZ until you encounter

Do you want to do a ROLLOVER run?
(Hit <CR> for default of no): Y

Note that we answer yes. BATGEN then responds with:

```
OBJECTIVE FUNCTIONS
RETRIEVED FROM MATRX.ROW
1.  OB1PNW20
2.  OB2TIM20

Please enter number of objective functions you wish to solve in this rollover run,
Maximum of 5:  2
```

Here you select the number of objective functions to be processed in the rollover run. In this case only 2 objective functions are possible because that is all that have been specified in the original CAC data and generated by the FORPLAN matrix generator. The next question is

```
1.  OB1PNW20
2.  OB2TIM20

Please enter the number of objective function 1 of 2:  2
```

The answer to this question identifies which objective function (OB2TIM20) is to be optimized first. BATGEN then asks

```
Would you like to:
1.  Maximize this function or
2.  Minimize this function:  1
```

Here, we wish to maximize OB2TIM20. The next question is

```
1.  OB1PNW20
2.  OB2TIM20

Please enter the number of objective function 2 of 2:  1
```

The answer to this question identifies which objective function (OB1PNW20) is to be optimized second. BATGEN then asks

```
Would you like to:
1.  Maximize this function or
2.  Minimize this function:  1
```

Here, we wish to maximize OB1PNW20. BATGEN then summarizes what we have specified for the rollover run and provides an opportunity to change our mind

```
You have entered these objective functions in the following order
OB2TIM20 MAX
OB1PNW20 MAX

ARE THESE OBJECTIVE FUNCTIONS CORRECT?
Y for YES, N for NO followed by a carriage return:  Y
```

The remaining questions are as described earlier unless you wish to make changes in the rollover related answers.

After you execute BATGEN your batch file and OBJ.FIL will be created. OBJ.FIL contains data on the objective functions to be optimized during the rollover run. For our example OBJ.FIL contains

```
2
OB2TIM20 MAX
OB1PNW20 MAX
```

This file is read by the utility ROLLOVER.EXE.

Because we are dealing with two objective functions in our example, our batch file will execute C-WHIZ twice (we do not reproduce the batch file here because of its size). The ROLLOVER utility coordinates the entire process and prepares the LP problem modification data needed by C-WHIZ. It creates and uses the following temporary files: OBJ.FIL, MATRX.MOD, ERROR.TMP, ROLLSTOP.TMP, and ROLL.NUM. The first time ROLLOVER is executed, it reads OBJ.FIL to determine the number of runs that are to be made (in our example, two) and the objective function information. C-WHIZ is then executed to solve the problem with the OB2TIM20 objective function. The problem is stored in the C-WHIZ file ACTFILE.ACT and SOLB2A writes out the first solution in SOL.FIL. ROLLOVER is executed again and it prepares the information needed to modify the problem for C-WHIZ. ROLLOVER writes this in the file MATRX.MOD and for our example it appears as follows:

```
NAME          FORPLAN
ROWS
   MODIFY
 G     OB2TIM20
RHS
   MODIFY
       RHS 1    OB2TIM20        98719.5469
ENDATA
```

MATRX.MOD is read by the MPSIN utility provided as part of the C-WHIZ package (see Ketron 1992 for more information on MPSIN). MPSIN uses this information to modify the original problem stored in ACTFILE.ACT. The modifications include changing OB2TIM20 from a free row to a greater than constraint and imposing a right-hand-side value (RHS) of 98719.5469 on that constraint (see below for discussion of where this number comes from). C-WHIZ is then executed again to solve the modified problem using the OB1PNW20 objective function, after which SOLB2A is executed to write out the new SOL.FIL. Note that the original MATRX.* LP data chapters are untouched during this process. Also note that if 3 or more objective functions are to be used, the above process, starting with the second execution of ROLLOVER, will be repeated as many times as there are additional objective functions.

After we execute our rollover batch file, three new files (LP.LOG, SOL.FIL, and LP.FIL) are created. LP.LOG contains timing and date of run information for both C-WHIZ runs, while SOL.FIL contains only the optimal solution resulting from the second C-WHIZ run. LP.FIL contains complete sets of information (matrix statistics, iteration logs, solution results, etc.) for both C-WHIZ runs.

Inspection of the two solutions in LP.FIL reveals the following about values for OB2TIM20 and OB1PNW20:

	OB1PNW20 (In Dollars)	OB2TIM20 (In MCF)
RUN 1	-21513.9762	99716.7103
RUN 2	-16368.7340	98719.5469

The value of OB2TIM20 dropped slightly from run 1 (where it was the objective function) to run 2 (where it was a constraint). The lower number is the value calculated by ROLLOVER (and shown above in MATRX.MOD) for the RHS to be used for OB2TIM20. The slight reduction is necessary to ensure feasibility for the modified problem because of the effects of numerical rounding errors. The negative PNV is increased from $21,513 to $16,368 in the second run. The optimal or "best" loss we could expect for the CAC model was -$15,990.75 (fig. 15).

No Feasible Solution

The most common problem encountered when solving FORPLAN models occurs when there is no feasible solution (i.e., no set of values for the decision variables exists that simultaneously satisfies all of the constraints in the model). Two or more constraints that cannot be simultaneously satisfied will result in no feasible solution. For example, the constraints x_1

< 3 and $x_1 > 10$ result in an infeasibility. In general, infeasibilities indicate that there are conflicting demands, limits, or inadequate production capabilities in the LP model. The problem cannot be solved as formulated.

Unfortunately, infeasibilities are frequently difficult to resolve. Their resolution requires determining which constraint or constraints are the source of the problem and then relaxing the offending constraint(s) by sufficient amounts to create feasible solutions. It is also possible that the model does not include an adequate range of production options (prescriptions or timing choices). Here we present two simple examples of infeasibilities occurring in the CAC data, showing how they were identified by C-WHIZ and discussing how they may be resolved. It is important to recognize that the resolution of infeasibilities in FORPLAN models is an art. A good discussion of this art was developed many years ago by Tom Stuart and may be found in Appendix VII of the *FORPLAN Version 1: User's Guide* (Kelly et al. 1986). Unfortunately, it is based on the use of FORPLAN Version 1 and the UNIVAC LP solver FMPS, so the specific details are out of date. If you wish to gain an understanding of the process for tracking down infeasibilities in FORPLAN models, it is still worth reading. One general piece of advice from this appendix is paraphrased here.

In sorting out the infeasibilities in a model, it is important to realize that the rows and/or columns listed as infeasible by C-WHIZ are simply those that were infeasible when C-WHIZ concluded there was no feasible solution to the problem. These rows and columns are not necessarily related to the row or rows that actually caused the problem to be infeasible. Thus there are two points to keep in mind:

1) You may have to search diligently and think hard to determine the factors that caused the problem to be infeasible.

2) In some cases it may be prudent to fix the infeasibilities you understand and then rerun the model hoping the infeasible rows you do not understand and did not fix were not the true cause of the problem.

The only advice we can add is to suggest that you keep a careful log of all changes you make to your model between runs. That way if you encounter an infeasible solution, you know that it must relate to the changes made since the last optimal run.

Before we turn our attention to the examples mentioned above, two points are in order. First, the CAC data set contains almost no constraints except for demand constraints; thus our examples are somewhat contrived. They will, however, give you some idea about how to use the information provided by C-WHIZ to help track down infeasibilities. Second, we use some specific FORPLAN terminology in discussing the examples. If some of the terms used are not familiar to you, refer to either the overview document (Johnson et al. 1986) or the user's guide (LMP 1992).

If you examine the CAC FORPLAN data (either in DATA.CAC or MAT.FIL) you will note that, for the 2,400 acres of analysis area number 037 not in allocation zones, a prescription control has been specified for the prescription with a code of TF. The control states that at least 45% (1,080 acres) and not more than 75% (1,800 acres) of this analysis area can be allocated to timing choices (12 columns in all) for the prescription coded TF.

The matrix generator creates a greater-than-or-equal-to row (PC037 1) to represent the lower limit of 45%. Figure 16 shows the right-hand-side (RHS) values for all rows in the CAC model with nonzero RHS's (file MATRX.RHS). Note that PC037 1 has a RHS of 1080. The upper limit for the control is incorporated into the model by specifying a range for PC037 1 in the file MATRX.RNG. Because there is only one ranged row in the CAC model, the only entry in the ranges file is

```
    RNG 1              PC037  1              720.
```

The first set of characters (RNG 1) defines the name of the ranges vector (printed in LP.FIL in the C-WHIZ input section), the second set specifies the row of interest, and the value is the width of the range (1,800 minus 1,080). The RHS and range values play key roles in our examples.

```
RHS
    RHS    1          LCNON              7840000.
    RHS    1          PC037  1              1080.
    RHS    1          AA037                 2400.
    RHS    1          AA052                 1000.
    RHS    1          AA883                 4500.
    RHS    1          AAS.0.                   1.
    RHS    1          AZ      20            2640.
    RHS    1          AZ      80            2800.
    RHS    1          AZ      90            2640.
    RHS    1          AZ     100            2800.
```

Figure 16.—Contents of MATRX.RHS for the CAC model.

There are two times during which C-WHIZ can detect infeasibilities: while PRESOLVE is running, or while WHIZARD is trying to solve the model. The following examples address each of these cases.

Infeasibilities Detected by PRESOLVE

PRESOLVE has the capability to detect certain types of infeasibilities as it analyzes the LP model matrix. Infeasibilities detected by PRESOLVE are sometimes referred to as patent or structural infeasibilities. If PRESOLVE detects an infeasibility, it will so indicate in LP.FIL. See Chapters 6 and 7 of Ketron (1992) for more details on interpreting PRESOLVE infeasibility information.

To demonstrate PRESOLVE's handling of a patent infeasibility, consider the following. Suppose you wish to modify the prescription control described above so that exactly 80 acres are allocated to prescription TF timing choices. You could make the necessary changes in DATA.CAC, regenerate the matrix, and solve the new LP model. For a small model, this process would be reasonably quick. Another approach, however, avoids regenerating the matrix by editing the existing MATRX.* files. For large models and simple changes, this approach is considerably quicker with a fast editor because it avoids the time-consuming matrix generation step.

Three editing changes need to be made to modify the model:

1) In MATRX.ROW, find the entry for row PC037 1 and change the type of constraint from G to E.

2) In MATRX.RHS (fig. 16), find the entry for row PC037 1 and change the RHS value from 1,080 to 80.

3) In MATRX.RNG, "comment out" the single entry by adding an * in the first column of that entry.

These changes make PC037 1 an equality row with a RHS of 80. In general, care must be taken when editing matrix files (and certain files such as MATRX.RX and MATRX.CAC should rarely be edited) to keep everything in the proper columns and in proper order. Refer to Chapter 6 of Kent et al. (1992) for more details on editing these files.

Next, suppose you execute an LP batch file (like LP.BAT described earlier in this chapter) and notice from the output printed to your screen that C-WHIZ has declared the problem infeasible. In general, if you have an infeasible model, information on your infeasibilities can be found in up to three places:

1) in LP.FIL,

2) in OOL.FIL, and

3) in your report (REPORT.FIL if you run the report writer — see Chapter 6).

Because the most complete information on infeasibilities is found in LP.FIL (if you know how to interpret it), we look there. Figure 17 shows the portions of LP.FIL that relate to the infeasibility. As before, sets of three vertical dots denote missing sections of the file.

First note that embedded in the PRESOLVE output is the message

***** INFEASIBILITY: ROW PC037 1(30)FORCES COLUMN 1037TF 1(168) BELOW BOUND**

This message indicates that PRESOLVE has found an infeasibility in the 30th row (named PC037 1), caused by the 168th column (named 1037TF 1), which has been forced below the value specified for its lower bound. In most cases the lower bound for a column will be zero because of the nonnegativity constraints.

Even though PRESOLVE has declared this problem infeasible, C-WHIZ enters the FAST PRIMAL solution procedure as evidenced by the iteration log in figure 17. This procedure reduces the sum of infeasibilities to as small a number as possible. Note the fact that at 31 iterations, XNIF = 1 (the number of infeasibilities) and XSIF = 80 (the sum of infeasibilities). These values mean that the WHIZARD optimizer was able to reduce the number of infeasibilities to 1 and the total sum of infeasibilities to 80.

Section 1 of the solution output (fig. 17) contains information on the nature of the infeasibility. First, row 30 (PC037 1) has been declared infeasible as denoted by ** in the status (or AT) column of the solution output. The slack activity, the lower limit, and the upper limit for this row are all equal to -80. The first question that comes to mind is — why are the lower and upper limits at -80? We expect these numbers to be the same because we made PC037 1 an equality constraint. However, we specified a RHS of +80, not -80. We check MATRX.RHS and discover that we had mistakenly typed in a -80, rather than 80, for the RHS for this row. The slack value of -80 is an artifact of WHIZARD's attempt to satisfy the constraint and is, ignoring sign, the source of the XSIF = 80 that we noted above.

The infeasibility arises because to satisfy the -80 RHS one or more of the timing choice columns for prescription TF must be driven to a negative value in solution, thus violating their respective nonnegativity constraints. The information on the solution for column 1037P1 1 is included in figure 17 to show what happened to the 2,400 acres on analysis area 037 in the infeasible solution. Finally, if we had made the change correctly in MATRX.RHS, the problem would have been feasible.

The important points to note are

1) PRESOLVE can detect and flag certain simple infeasibilities.

2) In the case of infeasibilities identified by PRESOLVE, C-WHIZ will still execute WHIZARD for the purpose of determining a solution that minimizes the sum of infeasibilities.

3) Information contained in LP.FIL can help track down the cause of the infeasibilities.

4) Often, patent infeasibilities arise from data input errors, either in your FORPLAN input data or in (as in the example) modifications to the LP data (see Appendix VII of Kelly et al. 1986 for examples relating to FORPLAN data).

5) This example is so simple that we were able to relate the sum of infeasibilities (XSIF) directly to the only problem causing the infeasibility. In practice, more than one source of infeasibility may exist, making the value of XSIF more difficult to interpret.

```
.
.
.
PRESOLVE:      4.68 SEC.
     1    COLUMNS TRANSLATED BY SINGLETON ROWS.
     6    DEGENERATE ROWS INTERSECTING       71  DEGENERATE COLUMNS ARE IGNORED.
     3    ALL NEGATIVE ROWS FREED.
***   INFEASIBILITY:  ROW PC037  1  (  30)  FORCES COLUMN 1037TF 1  (    168) BELOW BOUND
     4    COLUMNS TRANSLATED
    19    TRANSFER COLUMNS FREED.
   542    ACTIVE VECTORS IN THE MODEL.

ENTER FAST PRIMAL:      4.95 SEC.
     0    ITERATIONS        XFUNCT=18847.73019       XNIF=          25    XSIF=     5856029.9200

PI-WEIGHTING DISCONTINUED
INVERT DEMAND SET - INFEASIBLE

ENTER FAST PRIMAL:      5.06 SEC.
    31    ITERATIONS        XFUNCT= 17783.77245-     XNIF=           1    XSIF=          80.00000

INFEASIBLE SOLUTION:    5.12 SEC.
    31    ITERATIONS        XFUNCT=17783.77245-      XNIF=           1    XSIF=          80.00000

RESTORE ORIGINAL CONSTRAINTS

ENTER FAST PRIMAL:      5.12 SEC.
    31    ITERATIONS        XFUNCT=17783.77245-      XNIF=           1    XSIF=          80.00000

PI-WEIGHTING INVOKED
PI-WEIGHTING DISCONTINUED

INFEASIBLE SOLUTION:    5.12 SEC.
    31    ITERATIONS        XFUNCT=17783.77245-      XNIF=           1    XSIF=          80.00000

ENTER FAST PRIMAL:      5.17 SEC.
    31    ITERATIONS        XFUNCT=17783.77245-      XNIF=           1    XSIF=          80.00000

INFEASIBLE SOLUTION:    5.17 SEC.
    31    ITERATIONS        XFUNCT=17783.77245-      XNIF=           1    XSIF=          80.00000


SECTION 1 - ROWS

NUMBER . . . ROW. .     AT    . . ACTIVITY.     . . . .SLACK    . . . LOWER        . . . UPPER     . . . .DUAL
                                                  ACTIVITY.       LIMIT. . .         LIMIT. .     ACTIVITY.
       .
       .
       .
    30    PC037         1 **               .     -80.0000       -80.0000          -80.0000       -1.0000
       .
       .
       .
SECTION 2 - COLUMNS

NUMBER . COLUMN .       AT    . . ACTIVITY.       . . .INPUT    . . . LOWER       . . . . UPPER   . .REDUCED
                                                   COST. . .      LIMIT. . .         LIMIT. .       COST. . .
       .
       .
       .
   240    1037P1        1 BS    2400.0000         0.5448              .              NONE                .
       .
```

Figure 17.—Portions of LP.FIL for the infeasibility detected by PRESOLVE.

Infeasibilities Detected by WHIZARD

A second, far more common, and usually far more difficult class of infeasibilities to solve are those detected by WHIZARD while it is trying to solve a model. Infeasibilities of this type indicate that you have formulated a model that simply cannot be solved; i.e., you have specified a mix of goods and services that is beyond the capability of your production system (your forest) to provide.

Again we use the prescription control specified in the CAC data set to demonstrate how C-WHIZ handles this type of infeasibility. Suppose we wish to modify this control so that anywhere from 70% (1,680 acres) to 100% (all 2,400 acres) of analysis area 037 must be allocated to timing choices of the prescription coded TF. If we choose to modify our original LP data only one modification is required, and that is to change the RHS for row PC037 1 from 1,080 to 1,680 in MATRX.RHS. The constraint should still be a greater-than-or-equal-to type (so no change is needed in MATRX.ROW) and, by coincidence, 2,400 - 1,680 = 720, so no change is needed in MATRX.RNG.

Suppose next that we now attempt to solve the modified problem with C-WHIZ and notice from the output printed to the screen that C-WHIZ has declared the problem infeasible. As with the example above, there are potentially three places (if we run the FORPLAN report writer) to check for more information on the infeasible solution. As before, we refer to LP.FIL. The relevant portions are shown in figure 18.

No output from PRESOLVE has been included in the figure. It appears much as shown in figure 12 because no patent infeasibilities were detected. Looking at the iteration log in figure 18 we note that at iteration 30, WHIZARD declares the problem infeasible with one infeasibility (XNIF) and the sum of that infeasibility is equal to 280 (XSIF).

Looking next at the solution information in figure 18, we note that row PC037 1 has an activity level and lower limit of 2,680 and an upper limit of 3,400. Row AA037 (the analysis area acreage row for analysis area 037's nonzone acres) has activity level, lower limit, and upper limit all equal to 2,400. Both rows are binding and neither is flagged as infeasible. Looking at the columns we note that 1037TF 1 is in solution on 2,680 acres while 1037T1 1 is in solution on -280 acres and is flagged as infeasible because the non-negativity restriction is violated.

The first thing that stands out in this analysis is the activity level of 2,680 for the prescription control row and the one prescription column. Because only 2,400 acres exist on the analysis area, 280 excess acres appear. Checking MATRX.RHS where our only change was made, we find that we typed in 2,680 for the RHS of row PC037 1 instead of 1,680. This error explains all of the difficulties. The 3,400 upper limit on row PC037 1 comes from the range value in MATRX.RNG (3,400 = 2,680 + 720) and column 1037TF 1 is in solution at 2,680 acres to satisfy the prescription control row. Column 1037T1 1 is in solution at -280 acres to satisfy row AA037 (2,680 - 280 = 2,400), and finally, the sum of infeasibilities (XSIF) of 280 is the same 280 acres without regard to sign. Had the change in MATRX.RHS been made correctly, the problem would have been feasible.

The important points to note are

1) C-WHIZ can declare rows, columns, or both rows and columns to be infeasible.

2) As mentioned above, keep a careful log of model changes and be prepared for some hard work sleuthing infeasibilities.

3) This example is so simple that we were able to relate the sum of infeasibilities (XSIF) directly to the only problem causing the infeasibility. In practice, more than one source of infeasibility may exist, making the value of XSIF more difficult to interpret, and making the infeasibilities more difficult to resolve.

4) If all else fails and you are an agency analyst contact the LMP Systems Section for help by DG or at (303) 498-1833.

```
     .
     .
     .
ENTER FAST PRIMAL:      4.95 SEC.
    0   ITERATIONS      XFUNCT=18811.65739      XNIF=        25   XSIF=    5856229.9200

PI-WEIGHTING DISCONTINUED
INVERT DEMAND SET - INFEASIBLE

ENTER FAST PRIMAL:      5.17 SEC.
   30   ITERATIONS      XFUNCT= 16451.01028-    XNIF=         1   XSIF=       280.00000

INFEASIBLE SOLUTION:    5.17 SEC.
   30   ITERATIONS      XFUNCT= 16451.01028-    XNIF=         1   XSIF=       280.00000

RESTORE ORIGINAL CONSTRAINTS

ENTER FAST PRIMAL:      5.17 SEC.
   30   ITERATIONS      XFUNCT= 16451.01028-    XNIF=         1   XSIF=       280.00000

INFEASIBLE SOLUTION:    5.22 SEC.
   30   ITERATIONS      XFUNCT= 16451.01028-    XNIF=         1   XSIF=       280.00000

ENTER FAST PRIMAL:      5.22 SEC.
   30   ITERATIONS      XFUNCT= 16451.01028-    XNIF=         1   XSIF=       280.00000

PI-WEIGHTING INVOKED
PI-WEIGHTING DISCONTINUED

INFEASIBLE SOLUTION:    5.22 SEC.
   30   ITERATIONS      XFUNCT= 16451.01028-    XNIF=         1   XSIF=       280.00000


SECTION 1 - ROWS

NUMBER . . . ROW. .    AT   . . ACTIVITY.    . . . .SLACK   . . . LOWER      . . . UPPER    . . . .DUAL
                                               ACTIVITY.     LIMIT. . .       LIMIT. .   ACTIVITY.
     .
     .
     .
   30   PC037          1 LL    2680.0000              .     2680.0000      3400.0000       1.0000
   31   AA037          EQ      2400.0000              .     2400.0000      2400.0000      -1.0000
     .
     .
     .
SECTION 2 - COLUMNS

NUMBER . COLUMN .      AT   . . ACTIVITY.      . . .INPUT   . . . LOWER      . . . UPPER  . .REDUCED
                                                 COST. . .    LIMIT. . .       LIMIT. .     COST. . .
     .
     .
     .
  168   1037TF         1 BS    2680.0000         0.3903              .          NONE              .
  180   1037T1         1 **    -280.0000         0.4038              .          NONE              .
     .
     .
     .
```

Figure 18.—Portions of LP.FIL for the infeasibility detected by WHIZARD.

Efficiency Considerations

C-WHIZ is a very powerful and efficient LP solver. Our experience suggests that most models will be processed with little or no difficulty and on 80486-based computers in less (and often substantially less) than 2 hours. In the event of difficulties there are a number of factors to consider.

Rows

All rows except free rows are of concern. All efforts to keep the row count to a minimum will help reduce solution time. The use of intervals, where appropriate for constraints in the later periods is suggested. However, the implications for the optimal solution of doing this should be evaluated.

Coefficients

The larger the number of nonzero coefficients, or unique values of coefficients, the longer the solution time will be. Certain constraints such as demand constraints often add a significant number of coefficients to a model. One example is known where four forest-wide demand curves (defined for only the first five decades) nearly doubled the number of nonzero coefficients from 1.1 to 2.2 million.

Follow the advice given in Chapter 4 about reducing significant digits in FORPLAN-generated coefficients by using the appropriate FORPLAN data input sections (see the FORPLAN user's guide LMP 1992 for details).

Scaling

If C-WHIZ has difficulty with a poorly scaled matrix, it may be helpful to invoke the automatic scaling feature. See Ketron (1992) for details on invoking this feature of C-WHIZ and for a very good discussion on matrix scaling. If you have questions about scaling, contact the FORPLAN Hotline at (303) 498-1833.

Degeneracy

C-WHIZ has the ability to recognize and flag degenerate solutions. When it executes POSTSOLVE after optimization, the message "DEGENERACY MODE OFF" may appear, indicating that WHIZARD encountered degeneracy during the solution process and took defensive action. This message indicates that the effects of that action are removed. See Ketron (1992) for more details on degeneracy.

CHAPTER 6. GENERATING REPORTS

Once the LP model has been generated and successfully solved, solution reports can be generated. FORPLAN has the ability to report interpreted output from an optimized (or infeasible) matrix in four different ways. First, a standard report file (REPORT.FIL) can be generated that is comprised of a variety of default and optional reports formatted for a wide-carriage printer. Second, if a data file contains zones, zone reports can be generated (in the standard report file) that present solution results prorated by zone. Third, a standard flat file can be generated that contains the same output as the standard report file, but without the labels, headings, and graphs. Fourth, a relational flat file can be generated for some of the reports contained in the standard report file. More detail on reports and their use can be found in the Reports Chapter in the User's Guide (LMP 1992).

Both types of flat file reports can be loaded into relational data bases or other software that offer further reporting capabilities. Standard flat files were originally designed to interface with the Data General Present data reporting system. Analysts in Region 6 have developed the FLATALL software package, which allows standard flat files to interface with PARADOX, ORACLE, or other data base management systems (DBMS's). See Appendix B for information on using FLATALL. The relational flat file was originally designed to interface with the ORACLE DBMS. WO LMP Systems Section programmers have developed software to facilitate this process and it is described in Appendix C. Systems Section programmers have also developed the F2P utility to allow the relational flat file to be interfaced with PARADOX and this program is described in Appendix D.

Region 6 analysts have also developed a broad class of utilities that are designed to facilitate numerous FORPLAN data input and reporting tasks. These programs are briefly described in Appendix E.

NOTE: The standard FORPLAN reports and zone reports are formatted for 132-character-wide printers. If you plan to look at the reports on your computer, be sure that your viewing software is set up to view 132-character- wide documents. If you plan to print any portion of the reports, be sure that your printer and printer software are also set up for 132-character-wide documents. If you do not have a printer and software to handle 132-character-wide reports, you can export the report file to the DG and print the reports by using a landscape or compressed print command on a DG laser printer.

In order to produce reports, the FORPLAN report writer (RPTWTR.EXE) reads the LP model solution results (SOL.FIL — see Chapter 5 for a discussion of this file), and the FORPLAN input and yield data. The report writer prints information on LP solution results pertaining to rows and other columns (those contained in MATRX.OTH) similar to that contained in SOL.FIL. You may specify that these results be suppressed on the input data TITLE card as described in the Title Card chapter of the User's Guide (LMP 1992). Regardless of the format of the reports (standard, zone, flat file, or relational flat file) you wish to generate, the list of reports actually generated is user controlled in the FORPLAN input data via a data input section described in the Reports Chapter in the User's Guide (LMP 1992).

A. STANDARD REPORTS

The FORPLAN print report file (REPORT.FIL) contains forestwide information on all requested reports. We described the procedure to follow to produce standard reports using the BATGEN program in Chapter 3. In that case we used the SIX sample data set and produced the reports as part of a complete run.

Here we continue the example started in Chapter 4 (matrix generation) and continued in Chapter 5 (LP solution) where we use BATGEN and the CAC sample data. Again we assume that BATGEN is in your path and that the CAC data and yield data are in

C:\FORPLAN\CAC

as suggested in Chapter 2. Upon executing BATGEN, you will first see the introductory screen shown in figure 1. Select option 1, Runstream Generator. The next question asks for the name of the batch file to be created:

> **Enter runstream filename (a ".BAT" extension is required and is assumed if not typed in)**
> **(or hit <CR> for default of RUN.BAT):** REPORT

Note that our answer results in the file REPORT.BAT.

The next screen contains the list of possible run types (fig. 2) and we select **7. Report** for report generation. The next question is

> **Are there zones in the model? (y or n):** Y

We answer yes because the CAC data set contains zones. The next two questions ask for the path and file names for the data and yield files:

> **Enter data file name, including drive and path,**
> **(or hit <CR> for default of DATA.FIL:** DATA.CAC
>
> **Enter yield file name, including drive and path,**
> **(or hit <CR> for default of YIELD.FIL):** YIELD.CAC

Note that our answers presume that the data and yield files are in the same directory as REPORT.BAT. The next question is

> **REPORT TYPE**
> **1. Regular**
> **2. Zone**
> **Enter number of report type:** 1

We select the first option because we want to create regular or standard reports even though our data set contains zones. BATGEN's final question, as we have seen in previous chapters, asks if we wish to add another runstream to the batch file. For our purposes here, we answer no. BATGEN then terminates execution with a brief termination message as we saw in Chapter 3.

In the event that you have no file named SOL.FIL in your current directory, after you answer the question relating to report type, BATGEN will ask

> **SOL.FIL**
> **does not exist in the current directory**
> **Do you want to:**
> **1. Create SOL.FIL by running the LP solver**
> **2. Enter a SOL.FIL from another directory**
> **3. Exit**
> **Enter a 1, 2 or 3:**

This error check is necessary because, as noted above, a SOL.FIL must be present for REPORT.EXE to determine LP solution results.

Figure 19 shows a listing of the batch file that results from the BATGEN session just described. REPORT.BAT looks much like batch files discussed in earlier chapters with the usual remarks and file-related house cleaning occurring. Note the three copies of the data, yield, and solution files into various FORT.* files prior to execution of the report writer (RPTWTR.EXE). The report writer reads all three of these files, so all three must be present in the directory from which you are running or the report writer will abort. Consequently, BATGEN checks for all three and generates appropriate error messages if one or more are missing.

Also note the two copy commands after report writer execution. The first command creates the relational flat file (RDBDFLAT.FIL) and the second command creates the traditional flat file (FLAT.FIL). RDBFLAT.ZON exists if your data set has zones (see next section) while FORT.19 is present only if you request a relational flat file in your FORPLAN input data. On the other

```
echo off
Rem    Always start each run by deleting STATUS.TMP. This file is used to
Rem    communicate error messages between programs.
echo on
del status.tmp
Rem    REPORT WRITER STANDARD REPORT, 80387 VERSION
echo off
Rem    The file RDBFLAT.FIL is the new relational database flat file.
echo on
del report.fil
del yeildtbl.fil
del flat.fil
del rdbflat.fil
del fort.*
del report.log
copy YIELD.CAC fort.4
copy DATA.CAC fort.5
copy sol.fil fort.11
tim.exe  >report.log
rptwtr.exe <fort.5 >report.fil
tim.exe  >>report.log
copy rdbflat.zon+fort.19  rbdflat.fil
copy fort.20+fort.21+fort.22+fort.23+fort.24+fort.25+fort.26+fort.27+fort.28+fort.30  flat.fil
del fort.*

Rem    Sound Appropriate Tone at End of Run
if exist status.tmp goto :error1
beepend
goto :end1
:error1
beeper
:end1
echo End of Run #1
```

Figure 19.—Listing of REPORT.BAT.

hand, FLAT.FIL and all of the files used to build it are not created unless standard flat file reporting is requested in your input data.

After you execute REPORT.BAT, the following new files (size in bytes in parenthesis) will be added to your subdirectory:

REPORT.FIL (181,359)	This file contains the standard formatted output from the report writer for data sets with or without zones.
YIELDTBL.FIL (98,684)	This file contains the interpretation of the yield data by RPTWTR.EXE.
RDBFLAT.FIL (621)	This file contains only header information because a relational flat file was not requested in the CAC data.
REPORT.LOG (54)	This file contains the run date and starting and ending times (according to your computer clock) for execution of RPTWTR.EXE.

If errors occur during the execution of RPTWTR.EXE, error messages will be written beginning with a *E* in the first 3 columns of the report print file (REPORT.FIL) or in the yield table file (YIELDTBL.FIL) just as described in Chapter 4 for the matrix generator. Consult the User's Guide (LMP 1992) for a description of each error message. Execution of the report writer will continue even if errors are detected unless IFSTOP cards (LMP 1992) are encountered in the FORPLAN input data and the number of errors exceeds the number specified. The total report generation error count is printed at the end of the report print file at a "FINAL CHECK POINT."

B. ZONE REPORTS

If a data set contains zones, a zone file report (REPORT.FIL) that contains reports of the LP solution results prorated by zone can be generated. The zone report print file is similar to a standard report print file and is also produced by running RPTWTR.EXE. The original solution file (SOL.FIL) must be modified to create a ZONE.FIL prior to generating zone reports. This process is outlined in figure 20. Consult the User's Guide (LMP 1992) for a detailed description of zones and zone reports.

The zone report contains user-requested reports for each zone. They are produced in a two-step process described below:

STEP 1: The solution file (SOL.FIL), data file, and yield file are read and processed through RPTZ1, RPTZ2, and RPTZ3. RPTZ1.EXE reads the FORPLAN data and yield data, writes information to a series of temporary files, and writes out the rows and other columns portion of the ZONE.FIL. RPTZ2.EXE sorts the temporary files, and RPTZ3.EXE prorates the prescription allocations according to the portion of any analysis area in each zone and writes the relevant information to complete ZONE.FIL. RPTZ1.EXE writes data summary information and all three programs write error message and error check point information to ZON.FIL in wide-carriage print format. Consequently, if a zone file create run aborts, check ZON.FIL file for error messages.

STEP 2: The zone solution file (ZONE.FIL rather than SOL.FIL) created in step 1 is used by the report writer software along with the data file and yield file to create the zone report file (REPORT.FIL). The zone report file will contain zone totals, not forestwide information. Information printed for each zone is controlled by the user and may contain the CACs in solution, the analysis area prescription columns in solution prorated to the zones, and summary zone reports for each zone. Because RPTWTR.EXE processes all zones in ZONE.FIL, you may want to edit ZONE.FIL to remove zones you do not want reported before you create the zone report file. This process will reduce report processing time.

Using BATGEN to Create Zone Reports

BATGEN can be used to generate batch files to produce zone reports in two ways. In the first, two batch files are required; one for each of the two steps outlined above. Selecting run type 8, **Zonefile Create** (fig. 2), and answering questions relating to the location of your data provides BATGEN with the information it needs to construct the batch file shown in figure 21. Your FORPLAN input data, yield data, and relevant SOL.FIL must all be present in the subdirectory from which your batch file is being executed. RPTZ1.EXE, RPTZ2.EXE, and RPTZ3.EXE are all executed with their printed output and error message information going into ZON.FIL. While not obvious from the figure, the ZONE.FIL (modified SOL.FIL) is also produced. Execution of this batch file for our CAC sample data results in the creation of five new files (size in bytes in parentheses):

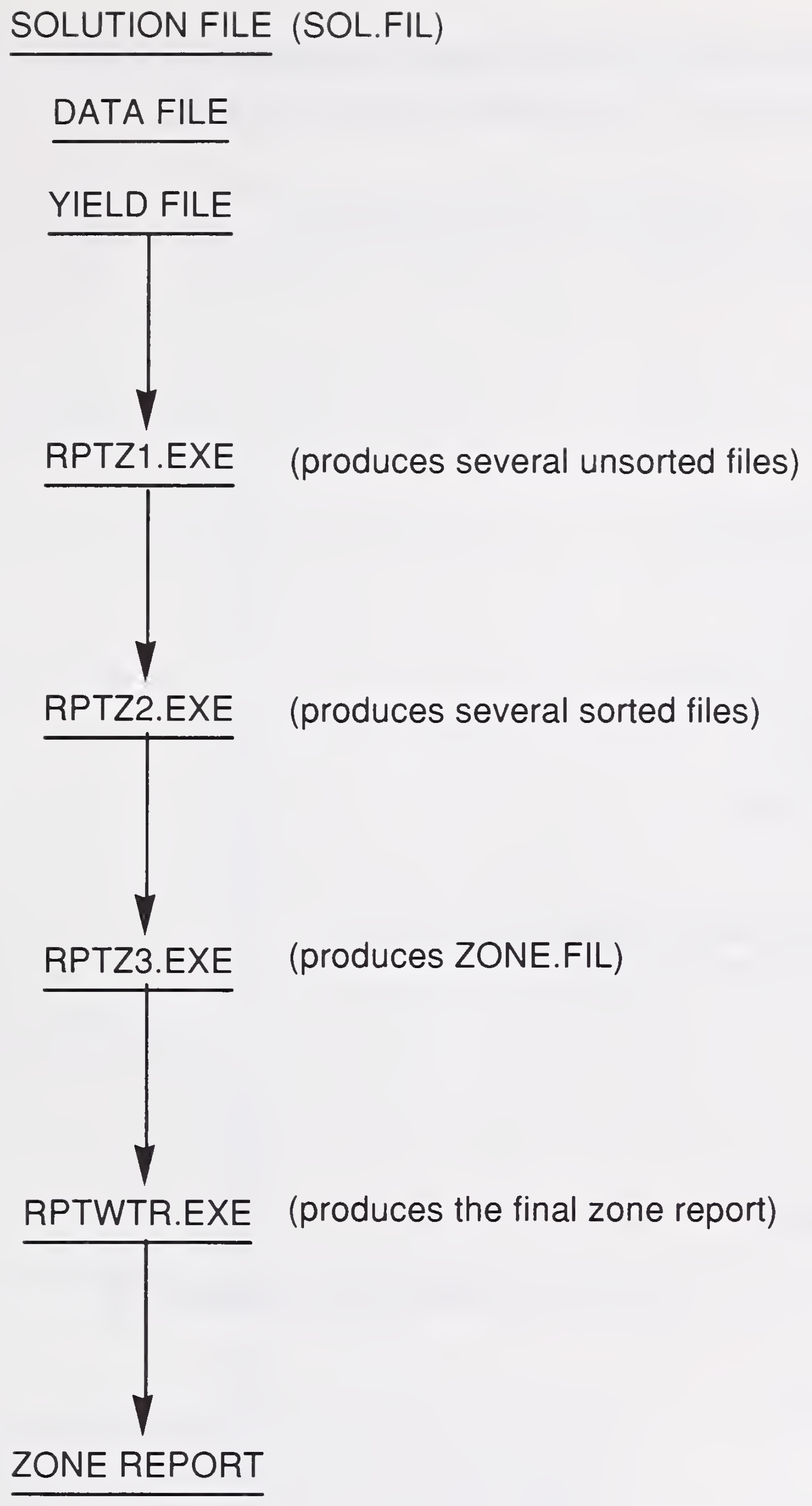
SOLUTION FILE (SOL.FIL)
DATA FILE
YIELD FILE
RPTZ1.EXE (produces several unsorted files)
RPTZ2.EXE (produces several sorted files)
RPTZ3.EXE (produces ZONE.FIL)
RPTWTR.EXE (produces the final zone report)
ZONE REPORT

Figure 20.—How zone reports are created.

YIELDTBL.FIL (112,542)	This file contains the interpretation of the yield data by RPTZ1.EXE and associated error messages, if any.
ZONE.LOG (54)	This file contains the run date and starting and ending times (according to your computer clock) for execution of the three RPTZx programs.
ZON.FIL (101,759)	This file contains output that summarizes FORPLAN input data, error messages, if any, and is produced by the three RPTZx programs.

```
echo off
Rem   Always start each run by deleting STATUS.TMP. This file is used to
Rem   communicate error messages between programs.
echo on
del status.tmp
Rem   ZONE FILE CREATION 80387 VERSION
del zon.fil
del yieldtbl.fil
del zone.fil
del rdbflat.zon
del fort.*
del zone.log
copy YIELD.CAC fort.4
copy DATA.CAC fort.5
copy sol.fil fort.11
tim.exe  >zone.log
rptz1.exe  <fort.5 >zon.fil
rptz2.exe  >>zon.fil
rptz3.exe  >>zon.fil
tim.exe  >>zone.log
del fort.*

Rem   Sound Appropriate Tone at End of Run
if exist status.tmp goto :error1
beepend
goto :end1
:error1
beeperr
:end1
echo End of Run #1
```

Figure 21.—Listing of zonefile create batch file.

RDBFLAT.ZON (620)	This file contains information pertaining to prescription columns in solution on acres within allocation zones, prorated across zones. This file becomes part of the relational flat file (RDBFLAT.FIL).
ZONE.FIL (16,749)	This file contains the modified solution file needed by the report writer to produce zone reports. Its contents are similar to those of SOL.FIL (see Chapter 5); all rows are included but only other columns (MATRX.OTH) in solution and prescription columns in solution for analysis areas in zones are included. For the prescription columns, acreage allocations are prorated across the allocation zones (fig. 22).

The second required batch file produces the zone report. Creating this file is similar to creating a standard report batch file with the only difference being your answer to the question:

```
REPORT TYPE
1. Regular
2. Zone
   Enter number of report type: 2
```

```
OPTIMAL   -15990.74902
          LCNON          EQ     7840000.00000     0.00000    7840000.00000  7840000.00000    0.00024
      .
      .
      .
          AZ     100     EQ        2800.00000     0.00000       2800.00000     2800.00000   -0.87129
ENDROW
          DEDO3 11       BS       18000.00320     0.02466          0.00000   200000.00000    0.00000
      .
      .
      .
          DEDO5 31       BS       16800.02400     0.00546          0.00000   147000.00000    0.00000
ZONE      END  START OF ZONE  20
          A   23   1              2640.0
          M012T2  1               1000.0
          M037T2  1                400.0
          M048TF  1                650.0
          M052TF  3                310.0
          M083GB  1                280.0
ZONE  20  END  START OF ZONE  80
          A   84   1              2800.0
          M012T2  1                670.0
          M012P1  1                230.0
          M037T2  1                150.0
          M037P1  1                250.0
          M048TF  1                650.0
          M048PF  1                100.0
          M083GB21                 570.0
          M092MN 1                 180.0
ZONE  80  END  START OF ZONE  90
          A   92   1              2640.0
          M012RD 1                1000.0
          M037RD 1                 400.0
          M048RD 1                 650.0
          M052RD 1                 310.0
          M083RD 1                 280.0
ZONE  90  END  START OF ZONE  100
          A   102  1              2800.0
          M012RD 1                 900.0
          M037RD 1                 400.0
          M048RD 1                 750.0
          M083RD 1                 570.0
          M092RD 1                 180.0
ENDALL 100
```

Figure 22.—Portions of ZONE.FIL for the CAC sample data.

Here we answer with a 2 to specify a zone report. BATGEN then checks for the presence of ZONE.FIL in the current directory. If present, the batch file needed to produce a zone report is created. This batch file will look like the one shown in figure 21 except that ZONE.FIL is used as input to the report writer instead of SOL.FIL.

Execution of this second batch file results in the creation of four new files (sizes in bytes in parentheses):

REPORT.LOG (54)	This file contains the run date and starting and ending times (according to your computer clock) for execution of RPTWTR.EXE.

YIELDTBL.FIL (98,684)	This file contains the interpretation of the yield data by RPTWTR.EXE and associated error messages, if any.
RDBFLAT.FIL (621)	This file contains the relational flat file associated with the zone report if one has been requested. If not, it will contain only the information from RBDFLAT.ZON produced by the RPTx programs. The latter case applies to the CAC data set.
REPORT.FIL (225,300)	This file contains the zone report printed in wide-carriage format.

Note: No standard flat file (FLAT.FIL) is produced because the CAC data set does not request one. If you wish, a standard flat file for zone reports can be produced by requesting it in your FORPLAN input data (LMP 1992).

While you are building a zone report batch file, BATGEN will check for a ZONE.FIL in your working directory. If none is found, BATGEN responds with

```
The file ZONE.FIL            does not exist.
Do you want to:
    1. Insert a Zonefile Create in this runstream?
    2. Enter a ZONE.FIL from another directory
    3. Request a Regular report instead of a Zone report
Enter a 1, 2 or 3:
```

Choose option one if you wish to do a zone report with a single batch file.

General Notes on Zone Reports

Following are a few notes on zone reports:

1) If an analysis area has only a portion of its acreage in zones, only those acres appear in the zone report.

2) It is important to realize that the zone report prorates the solution back into the zones. While the proration is done carefully, it is only a proration. A different result may be achieved by allocating analysis area acres in a manner other than straight proration.

3) Zone reports do not contain forestwide total information. This information must be obtained by executing a standard FORPLAN report run. We suggest you obtain a standard report first and then a zone report if you feel it is needed.

4) The ZON.FIL contains output from the creation of the zone file (ZONE.FIL). Check ZON.FIL file to be sure your zone file was created properly. Look for error messages that begin with *E* in the first three columns. Also, there are intermediate and final check points printed throughout the file. If problems arose during the creation of the zone file, information in ZON.FIL will assist you in analyzing them.

5) Zone reports cannot report across a collection of zones. That is, you cannot aggregate two or more zones and obtain a report on the aggregation. If you want to report across zones, you must add the information manually or use either the relational flat file or the standard flat file capability with additional software, such as a data base manager.

6) Using the zone reporting system allows you to circumvent some report restrictions present in the FORPLAN report writer. For example, the report writer limits you to a maximum of 200 multiperiod reports per execution. If you use the two-step zone-reporting process, you are limited to a maximum of 200 multiperiod reports per zone.

NOTE: An alternative to creating zone reports using FORPLAN is to create a relational flat file (RDBFLAT.FIL) and use a relational data base package such as ORACLE or PARADOX to create reports by zone.

C. FLAT FILES

The report writer software can create a flat file free of typical formatting conventions that can be used with other software. FORPLAN can produce two types of flat files — standard and relational. There are many reasons you might want to create a flat file including

- — to produce visually appealing reports
- — to produce attractive graphic displays
- — to move data into a data base or spreadsheet
- — to communicate with spatial software
- — to summarize your solution based on any combination of parameters (level identifiers, management strategies, treatments, etc.)

Creating Flat Files

Reports are not automatically written to a flat file (LMP 1992). You must request this procedure in the "Report Information to Be Printed" section of the input data. Entering an **F** or **B** in any of the optional report fields will trigger information to be sent to a standard flat file (FLAT.FIL). An **F** means a report should be printed to the standard flat file, but not to the report file. A **B** means a report should be printed to both the standard flat file and the report file. An **R** means that a report should be printed to the relational flat file (RDBFLAT.FIL). You cannot request standard flat file information and relational flat file information in the same report writer run.

Standard Flat Files

The standard flat file prints the report information in a free format file. The following report information can be written to a standard flat file:

- — activity/output reports
- — multiperiod optional reports
- — single-period optional reports
- — treatment-type reports
- — treatment-type optional reports
- — inventory, harvest, and growth report
- — inventory, harvest, and growth optional report
- — the age-class report
- — the age-class optional report

This data is contained in sets of five records:

Record 1 —	Lists basic information such as report type, activity or output, etc.
Record 2 —	Contains prescription column information. This record is blank except when information is requested at the column level.
Record 3 —	Contains information on identifiers, treatment types, and qualifiers. This record is blank unless reports were requested by identifiers, treatment types, or qualifiers.

Record 4 — Contains the qualifier of the quantity reported. (e.g., amount or dollar) and the first 10 period values.

Record 5 — Contains the qualifier of the quantity reported, (e.g., amount or dollar) and the last 10 period values.

NOTE: If the flat file has been reformatted from 132 to 80 characters in width, using a utility described in Appendix E, then the following applies:

1. Information on the third record will be split between the third and fourth record.
2. Information on the fourth record will be split between a fifth and sixth record.
3. Information on the fifth record will be split between a seventh and eighth record.

Instructions on loading standard flat files into relational data bases and performing analysis with these data are contained in Appendix B.

Relational Flat Files

The relational flat file allows you to extract basic information from the FORPLAN solution results using the report writer, and to write it in a relational data base format. Relational data base software products can be used to perform further analysis, aggregation, summation, report formatting, and graphics. The relational flat file will only accept prescription level information, and only from multiperiod optional reports (LMP 1992). To create a relational flat file

1. In the "Report Information to be Printed" section of the FORPLAN input data, place an "R" in the field for multiperiod reports on the prescription report selection card(s).
2. Complete the "Multiperiod optional report" data input section, observing the following rules:
    a. You cannot use activity or output aggregates.
    b. You can only request the amount per period.
    c. You cannot specify level identifiers, treatment types, or qualifiers.

The relational flat file information is located in RDBFLAT.FIL, and contains the following:

Section 1: This section is produced only if a data set has a zone formulation. It contains the proportion of each prescription column in solution on acres within allocation zones, allocated to each zone.

Section 2: This section contains model-specific information. It lists the temporal data, level identifiers, activities, outputs, qualifiers, treatment types, and analysis area information. If the data set has a zone formulation, it will also contain the zone and coordinated allocation choice information.

Section 3: This section contains the actual solution information. For each prescription on each analysis area it lists the amount of each output or activity that occurs in each time period.

Section 1 is produced automatically for any data set containing zones, even if a relational data base file is not requested. Section 3 contains the following information on each user specified activity and output.

1. For activities or outputs with yields either contained in the yield file or directly entered, Section 3 contains information on amount, dollar value, user defined qualifier, and treatment type.

2. For activities or outputs with yields derived from dependent relationships, Section 3 contains information on amount, dollar value, and treatment type.

For timber outputs, it is suggested that you maintain an explicit distinction between the timber being "produced" (cut) and the inventory on the ground by incorporating a separate inventory output in your model. The yield for this output can be derived internally through the use of yield composites or the yields can be explicitly entered in a yield table. The age attribute is only associated with inventory outputs. The relational flat file recognizes an inventory output as the one specified in the "PERPETUAL TIMBER HARVEST" section of the input data.

A relational flat file can be transferred to relational data base software. You may summarize your solution based on any combination of parameters (e.g., level identifiers, management strategy, and treatments). Two applications have been developed, one for ORACLE on the DG, and one for Paradox on microcomputers. These are discussed in Appendices C and D.

LITERATURE CITED

Ager, A.; Kent, B.; Merzenich, J.; Phillips, R. 1991. Application of Rocky Mountain Station Microcomputer FORPLAN on National Forests in the Pacific Northwest Region. In: Proceedings of the Symposium on Systems Analysis in Forest Resources; 1991 March 3-6; Charleston, SC. Gen. Tech. Rep. SE-74. Charleston, SC: U.S. Department of Agriculture, Forest Service, Southeast Forest Experiment Station: 247-249.

Bailey, R.G. (Tech Coord). 1986. Proceedings of the Workshop on Lessons from Using FORPLAN; 1986 April 29-May 1; Denver, CO. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Land Management Planning Systems Section. 268 p.

Hoekstra, Thomas W.; Dyer, A.A.; LeMaster, Dennis C. 1987. FORPLAN: An Evaluation of a Forest Planning Tool. Gen. Tech. Rep. RM-140. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Rocky Mountain Forest and Range Experiment Station. 164 p.

Iverson, David C.; Alston, Richard M. 1986. The Genesis of FORPLAN: A Historical and Analytical Review of Forest Service Planning Models. Gen. Tech. Rep. INT-214. Ogden, UT: U.S. Department of Agriculture, Forest Service, Intermountain Research Station. 31 p.

Johnson, K. Norman. 1986. FORPLAN Version 1: An Overview. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Land Management Planning Systems Section. 83 p.

Johnson, K. Norman; Stuart, Thomas W. 1987. FORPLAN Version 2: Mathematical Programmer's Guide. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Land Management Planning Systems Section. 123 p.

Johnson, K. Norman; Stuart, Thomas W.; Crim, Sarah A. 1986. FORPLAN Version 2: An Overview. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Land Management Planning Systems Section. 66 p.

Johnson, K. Norman; Scheurman, H. Lynn. 1977. Techniques for Prescribing Optimal Timber Harvest and Investment Under Different Objectives — Discussion and Synthesis. Forest Science Monograph 18. 31 p.

Kelly, James W.; Kent, Brian M.; Johnson, K. Norman; Jones, David B. 1986. FORPLAN Version 1: User's Guide. Washington, D.C.: U.S. Department of Agriculture, Forest Service, Land Management Planning Systems Section. 175 p.

Kent, Brian M. 1989. Forest Service Land Management Planners' Introduction to Linear Programming. Gen. Tech. Rep. RM-173. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Rocky Mountain Forest and Range Experiment Station. 36 p.

Kent, Brian M.; Sleavin, Katherine; Ager, Alan; Baltic, Tony. 1992. Operations Guide for FORPLAN on Microcomputers, Release 13. Gen. Tech. Rep. RM-219. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Rocky Mountain Forest and Range Experiment Station. 67 p.

Ketron. 1992. C-WHIZ Linear Programming Optimizer. Arlington, VA: Ketron Management Science Corp. 59 p.

LMP (Land Management Planning). 1992. FORPLAN Version 2: User's Guide, Release 14.2. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, and Management Planning Systems Section. 276 p.

Schrage, Linus. 1986. Linear, Integer, and Quadratic Programming With LINDO. Redwood City, CA: The Scientific Press. 284 p.

Schrage, Linus. 1989. User's Manual: Linear, Integer, and Quadratic ProgrammingWith LINDO. Redwood City, CA: The Scientific Press. 95 p.

APPENDIX A. LIST OF WEITEK DISTRIBUTION PACKAGE FILES

Following is a list of the files contained in the Weitek math coprocessor version of FORPLAN. Refer to Chapter 2 for details on downloading this package from the DG and on the nature of the files. Many of the executable programs have the same name as their Intel version counterparts. If you plan to maintain both versions on your system, keep them in separate directories.

Files contained in FPEXE_W.EXE are

ALL-L.BAT
ALL-W.BAT
C-WHIZ12.BAT
GEN.BAT
LINDO.BAT
PDO.BAT
REPORT.BAT
MATGEN.EXE
BEEPEND.EXE
BEEPERR.EXE
LMPSFX.EXE
LSOLFX.EXE
MATPDO.EXE
RPTWTR.EXE
TIM.EXE
BATFILES.HLP
LPINP.DAT
DATA.SIX
YIELD.SIX
BATGEN.EXE
ROLLOVER.EXE

Files contained in ZONE_W.EXE are

ALLZON-L.BAT
ALLZON-W.BAT
ZONEFILE.BAT
ZONEGEN.BAT
ZONEPDO.BAT
ZONERPT.BAT
MATGENM1.EXE
MATPDOM1.EXE
RPTZ1.EXE
RPTZ2.EXE
RPTZ3.EXE
DATA.CAC
YIELD.CAC

APPENDIX B. CREATING DELIMITED FLATFILES FOR DATA BASE ANALYSIS

The FLATALL program converts data stored in standard FORPLAN flat files into a format that can be read by other software. The delimited data files produced by FLATALL can be imported into ORACLE, PC PARADOX, or other data base or spreadsheet systems. The advantages of using FLATALL are

1) Each standard FORPLAN flat file record is converted from a five-line card image format to a single data line.
2) Records are put into a standard format with consistent names for each field.
3) A separate record can be generated for each period in which an output occurs. This procedure allows data analysis by time period.

FLATALL was designed to process standard flat files, not relational flat files. With FORPLAN, Release 14, you now have an option of directly generating relational flat files (see Chapter 6).

Complete instructions on building FORPLAN flat files and running the FLATALL program are contained in the file FLATFILE.DOC, which is distributed with the program. The following section summarizes the steps required to run FLATALL.

A. STEP 1: GENERATING FORPLAN STANDARD FLAT FILES

The FLATALL program reads and converts the following standard flat file information:

1) Forestwide activity/output summary data
2) Multiperiod optional reports
3) Single period reports (prescription level only)
4) Treatment type reports
5) Optional treatment type reports

By linking prescription-level FORPLAN solution data to yield data and harvest-timing choice data, timber inventory and growth information can be obtained. A procedure has also been developed to display the detailed forest age class structure at the end of each time period.

B. STEP 2: OBTAINING FLATALL.EXE

To obtain the program, RIS the file FLATFILE.ZIP from the following location:

R06A:STAFF:PLAN:MERZ:PROGRAMS:

Move this file to your microcomputer and "unzip" it by typing in the command >PKUNZIP FLATFILE. (**Instructions on loading the PKZIP and PKUNZIP programs onto your PC are contained in Appendix E.**) FLATFILE.ZIP contains

1) The executable program FLATALL.EXE.
2) A documentation file named FLATFILE.DOC.
3) Paradox database templates for each of the four flat file types.
4) A "field expansion" program named FLATEXP.EXE that works with single-period, multiperiod, and treatment-type reports.

FLATEXP.EXE can be used to expand the 2-character level identifier codes in the FLATALL output into either a 6- or 20-character name. Before running this program, load the FORPLAN identifier data into a separate file. The 6-character code is read as columns 7 to 14 and the 20-character code is read as columns 16 to 35. To create the Paradox structure files, copy the codes for the FLATALL output and restructure the level identifier fields from A2 to either A6 or A20.

For further assistance with these files and programs contact Jim Merzenich (J.MERZENICH:R06A) at (503) 326-5191. Two special versions of FLATALL are also available:

1) FLATBOB.EXE is a PC version not requiring a math coprocessor.
2) FLAT20P.EXE is a version designed to provide data for 20 periods.

C. STEP 3: RUNNING FLATALL.EXE

The FLATALL program is interactive. All input and output file assignments are based on user-supplied answers. To execute FLATALL, type

FLATALL

The program will ask the following questions:

1) Do you want to extract single-period, multiperiod, treatment-type, or forestwide activity/output data?
2) The names of the activities and/or outputs you want to extract. A maximum of 10 can be extracted; a blank response infers "all."
3) What report type do you want data for — forestwide, by zone, by analysis area, or by prescription?
4) The periods you wish data for (this answer may include a starting and an ending period).
5) The name of your flat file.

The output files produced from FLATALL are in fixed comma-delimited formats and have a .txt DOS file extension. The .txt files can be imported directly into spreadsheets or Paradox. Paradox data base templates, containing the exact structure of each file produced by FLATALL, have been built and are described below.

FA020.DB	Forestwide activity/output data (one record/output/period)
FA021.DB	Forestwide activity/output data (15 decades data per record)
FSINGLE.DB	Single-period report data
FMULT1.DB	"Detailed" multiperiod data
FMULT2.DB	"Condensed" multiperiod data (or RX activity/output summary data)
FMULT3.DB	"In-between" multiperiod data
FTREAT.DB	Treatment-type data

D. REPORT TYPES

Activity/Output summary reports — These reports contain summary data for every activity and output in the FORPLAN model. Normally this report is requested at the "forest" level. If this report is requested at the prescription level, a very big flat file will (reportedly) be produced containing the complete FORPLAN solution. We have not been brave enough to try this option.

Single-period reports — A single-period report contains data for one time period only. FLATALL processes reports for up to 10 outputs at a time.

Multiperiod reports — FLATALL will process multiperiod reports for up to 10 outputs at a time. If your flat file contains more than 10 outputs, make separate FLATALL runs for each set of 10 outputs. Running FLATALL using the "condensed" data option reduces the size of the output file.

Treatment-type reports — Treatment-type reports provide a useful method of describing timber harvests, because both harvest methods and volumes are contained on the same records.

When you request standard treatment-type information for the flat file, data are generated for all treatment types and their aggregates. Be aware that both raw and aggregated data will be in the flat file. When you request optional treatment-type information for the flat file, a separate report request must be made for each treatment type.

E. SPECIAL FLAT FILE REPORTS

Two special single-period flat file reports that will help you interpret your FORPLAN solution are worth noting.

Acres flat file — A flat file containing the existing and regenerated timing choices for all prescription columns in solution can be obtained by requesting a single-period report for period 1 on your "acre" output. This flat file is the key to understanding your model. You can use it to determine the prescriptions applied to an area and the harvest timing choices selected. This flat file can be used to summarize land allocations and harvest scheduling across any possible combination of level identifiers.

When building this flat file, FLATALL will also calculate stand ages associated with each prescription column in each period. These data characterize the overall forest structure through time.

NOTE: Because stand age is normally assigned from the timber yield file, most FORPLAN models do not assign ages to areas allocated to nontimber prescriptions. FLATALL uses a supplemental file containing beginning ages themed to level identifiers to assign ages to areas that are not harvested. You must create this supplemental file in the following format:

columns 1-16	level 1-8 identifier codes (do not use aggregate codes)	8A2
columns 17-18	beginning age (in decades)	I2

You may be interested in comparing the prescriptions in solution to the set of possible prescriptions. Because FORPLAN input data is normally "themed" using aggregates, it is often difficult to compare the prescription column data in solution with the input data. The program THEMEX.EXE decodes the "prescription source" and "harvest source" portions of the FORPLAN input data, making these direct comparisons possible.

LTSY flat file — To obtain an acres flatfile that applies to timber prescriptions only and also includes LTSY, request single-period, prescription-level flatfile output on LTSY for last decade in your planning horizon.

F. OUTPUT VARIABLES AND NAMES

This section contains the variables and output names used in three of the most commonly used data bases.

Single-Period Data: **FSINGLE.DB** (note that in this file age data is given for decades 1 through 15. A special version of the program FLAT20P.EXE that makes calculations for 20 periods is available from Jim Merzenich [R06A]).

Field	*Name*	*Format*	*Description*
1	COL	A8	Column name
2	ZONE	A4	Zone
3	OUTPUT	A8	Output/activity name
4	Amount	N	Amount of acres, volumes
5	Acres	N	Acres
6	Period	N	Time period (decades)
7-14	LEV1..LEV8	8A2	Level 1-8 identifiers
15	POI	N	Period of implementation
16	Begin Age	N	Beginning age
17	E Entry Age	N	Existing entry age
18	E Harvest Age	N	Existing harvest age
19	R Entry Age	N	Regen entry age
20	R Harvest Age	N	Regen harvest age
21	E Entry Per	N	Existing entry period
22	E Harvest Per	N	Existing harvest period
23	TNT	A1	T=timber, N=nontimber
24-38	Age01..15	N	Age at end of decade 1-15

Multiperiod Data: **FMULT1.DB**

Field	*Name*	*Format*	*Description*
1	AA	A4	Analysis area
2	RX	A5	Prescription name
3	CAC	A5	Coord. allocation choice
4	COL	A8	Column name
5	ZONE	A4	Zone
6	Acres	N	Acres
7-14	LEV1-LEV8	8A2	Level 1-8 identifiers
15	Period	N	Period output is produced
16	Amount	N	Output quantity (area)
17	Output	A4	Output/activity name

Treatment-type Data: **FTREAT.DB**

Field	*Name*	*Format*	*Description*
1	AA	A4	Analysis area
2	RX	A5	Prescription name
3	CAC	A5	Coord. allocation choice
4	COL	A8	Column name
5	ZONE	A4	Zone
6-13	LEV1-LEV8	8A2	Level 1-8 identifiers
14	Period	N	Period of treatment
15	Age	N	Age of stand treated
16	Treat	A1	Treatment type code
17	Volume	N	Harvest vol. (area)
18	Acres	N	Acres of treatment
19	Output	A4	Output name
20	Qual1	A1	Qualifier code
21	DBH	N	D.B.H.
22	RXTYPE	A5	EXIST or REGEN

APPENDIX C. CREATING REPORTS WITH ORACLE

FORPLAN has the ability to create relational flat files. These flat files can be loaded into a DG-based ORACLE data base. You may query the data base for solution information and design your own reports. The WO-LMP unit at Fort Collins has developed a set of macros that will create the ORACLE tables and load your flat file into the data base. In addition, sample views, queries, and query-by-examples are also provided.

To obtain a copy of the macros and the examples, RIS the following file:

Host:	**W04A**
Staff:	**LMP**
Drawer:	**FORPLAN**
Folder:	**ORACLE**
File:	**ORACLE.DMP**

Once the file is on your DG host, you may "load" it. Use IS main menu option "3. Utilities" for a loading menu. Following are the instructions to use the system.

A. STEP 1: CREATING THE ORACLE TABLES

The macro FORPLAN_TABLES.SQL contains a command file to create the ORACLE tables from the FORPLAN relational flat file data. It should be copied into the directory from which you access SQLPLUS. To execute this macro, type the following from the SQLPLUS command line:

START FORPLAN_TABLES

A complete description of each table (column names, format) can be found in the file FORPLAN_TABLES_DESCRIPTION.TXT provided in the ORACLE.DMP file. The tables created at this step are listed below.

```
AA_ID                  FP_THEME_CLASSES
AO_OCCURRENCES         FP_TRAITS
FP_ACTIVITIES          FP_TREATMENT_TYPES
FP_ANALYSIS_AREAS      FP_YEAR_GROUPS
FP_CACS                FP_ZONES
FP_COLUMNS             RX_ID
P_OUTPUTS              TEMP_AA_RX
FP_PRESCRIPTIONS       ZN_AA
FP_THEMES
```

Once this FORPLAN_TABLES.SQL is executed, it is no longer needed and may be deleted from your system.

B. STEP 2: LOADING DATA INTO ORACLE

The relational flat file must be moved to the DG and stored where the ORACLE macro files can access it. This relational file must be named RDB_FLAT.DAT. It can be a large file, so you may want to delete it once it is loaded into ORACLE. A small, sample flat file called RDB_FLAT is contained in the ORACLE dump file. You may wish to experiment with it before loading a larger file.

Once a relational flat file resides on the DG, two utilities must be executed to load it into the ORACLE tables created in Step 1. The utilities perform the following tasks:

1. Load the relational flat file into the tables
2. Transfer information from a temporary table (TEMP_AA_RX) into a permanent one.

1. Load the flat file into Oracle — RDB_FLAT.DAT should be moved into the directory from which you access SQLPLUS. SQL*LOADER is used to load this flat file into ORACLE tables. The file RPTLOAD.CTL is a macro file developed to accomplish this task and should reside in the same directory as the flat file. To invoke this procedure from the command line type

SQLLOAD / CONTROL=RPTLOAD

(the '/' is the auto logon symbol so leave spaces on either side)

NOTE: The loader deletes everything in the tables before loading new data. The loader will write out intermediate completion points to the screen and three diagnostic files. A discard file, RPT.DISC, contains records that did not meet any of the load criteria. A reject file, RPT.BAD, contains records that had problems loading. This file should be empty; if it is not, call the FORPLAN hotline. Finally, a record of the load procedure is contained in RPTLOAD.LOG .

2. Transfer information from temporary tables to permanent tables — In the previous step some of the data is loaded into a temporary table (TEMP_AA_RX). This data is processed again and inserted into permanent tables by a SQLPLUS query contained in the macro file RPTLOAD.SQL. This macro should be copied into the directory from which you access SQLPLUS. To execute the macro, type the following from the SQLPLUS command line:

START RPTLOAD;

NOTE: the SQLPLUS query will write out intermediate completion points to the screen.

C. STEP 3: CREATING ORACLE VIEWS

The file FORPLAN_VIEWS.SQL is a macro that will create necessary ORACLE views. The macro should be copied into the directory from which you access SQLPLUS. To execute this macro type the following from the SQLPLUS command line:

START FORPLAN_VIEWS

A complete description of the views (column names, format) is found in the file FORPLAN_VIEWS_DESCRIPTION.TXT. The views created at this step are listed below.

AA_RX	Joins analysis area and prescription information.
AA_TABLE	Creates a table of analysis area identifiers.
AO	Joins activity and output information.
AO_TABLE	Creates a table of activities and outputs by year group.
IDENTIFIERS	Joins theme and identifier information.
RX_TABLE	Creates a table of prescription identifiers.

Once the macro (FORPLAN_VIEWS.SQL) is executed it is no longer needed and can be deleted from your system.

D. STEP 4: SAMPLE QUERIES

The following files should be copied into the directory from which you access SQLPLUS. To execute a query, type the following from the SQLPLUS command line:

START query (eg. START ID)

The possible queries include:

ID.SQL	This query summarizes the FORPLAN identifiers.
AA_TABLE.SQL RX_TABLE.SQL	These queries describe the level identifiers used to define analysis areas and prescriptions.
AA_ON_RX.SQL RX_ON_AA.SQL	These queries summarize the combination of analysis areas and prescriptions. The first sorts analysis areas by prescription; the second sorts prescriptions by analysis areas.
ALLOCATION.SQL	This query allows the user to summarize the land allocation by any grouping of themes.
AO_80.SQL AO_132.SQL	These queries summarize forestwide activities and outputs (amounts and dollars) by planning period. The only difference between the two is the width of the output.
TT_80.SQL TT_132.SQL	These queries provide the acres of each treatment type by planning period for every relevant activity/output as well as the amount of the activity/output produced. The only difference between the two queries is the width of the output.
TRAIT.SQL	This query allows the user to break a particular trait (qualifier) into five groups and summarize output for each group.

E. QUERIES SIMULATING QUERY-BY-EXAMPLE

The file FORPLAN_QBE_TABLES.SQL contains a macro that will create the ORACLE query-by-example (+QBE) tables (AA_QUERY, RX_QUERY, and AO_QUERY). To execute the macro type the following at the SQLPLUS command line:

START FORPLAN_QBE_TABLES

After the QBE tables have been created, the following macros can be used to create reports and extract data.

FORPLAN_QBE.FRM	File containing SQL*FORMS application needed to run the +QBE queries. Analysis area and prescription criteria are located on the first page of the application; activity/output and trait information are loaded on the second page.
QBE.SQL	Command file needed to run the +QBE queries. It is called by all +QBE queries.
AO_80_QBE.SQL AO_132_QBE.SQL TT_80_QBE.SQL TT_132_QBE.SQL	These queries function like those described above except the user is able to specify criteria to refine the output. The criteria used include level identifier information, individual activities/outputs, and particular traits. The user indicates the specific criteria on an SQL*FORMS and should be familiar with SQL*FORMS because the application offers little assistance.
TRAIT_QBE.SQL	This query allows the user to break a particular qualifier into five groups (traits) and summarizes output for those traits based on QBE criteria.

APPENDIX D. CREATING REPORTS WITH PARADOX

A. WHAT IS F2P?

The WO-LMP systems staff at Fort Collins has developed a utility called F2P to move the FORPLAN relational flat file data into the Paradox data base system. F2P will create PARADOX data base tables and PARADOX primary index files, and load FORPLAN relational flat file information into these tables and files. The data base can then be queried to obtain information on the solution. F2P is a quick and easy way to link the power of FORPLAN with the power of a relational data base management system. You must have purchased a copy of Borland's PARADOX to use the data base tables created from F2P. If you have questions or encounter problems with running F2P, contact

K.Sleavin:W04A (303) 498-1833

B. HOW TO OBTAIN F2P

F2P can be RISed from:

W04A:STAFF:LMP:FORPLAN:UTILITIES:F2PZIP.EXE

F2PZIP.EXE is a self-extracting zip file. Simply RIS F2PZIP to your DG and transfer it to your PC. Once it is on your PC type in

```
F2PZIP
```

and it will "unzip" itself. Four files will be produced: F2P.EXE, which is the executable file; README.DOC, which is a duplicate of this announcement, and two sample relational flat files, RDB.FLAT and RDB.FLAT1. Once F2PZIP.EXE has been successfully unzipped, it may be deleted.

You should place the path to F2P in your AUTOEXEC.BAT file so it can be executed from any subdirectory.

If you do not have a relational flat file you may use one of the two provided (RDB.FLAT or RDB.FLAT1) to run F2P for learning exercises.

C. HOW TO EXECUTE F2P

To run F2P, simply enter the following at the DOS prompt:

```
F2P <flat_file>
```

where <flat_file> is the name of the FORPLAN relational flat file. First, F2P creates a new data base and primary index files, a summary of which is listed below. When F2P is executed, the screen will appear as follows:

```
//
//
/////// Creating and Opening the Paradox Tables
//
//
/////// Adding the Primary Keys to each Table
//
//
/////// Beginning File Translation
//
//
/////// Your Paradox '*.db' and '*.px' files are ready!
```

D. FILES CREATED BY F2P

F2P creates and opens the following Paradox tables (.DB files). Key fields are flagged with a **:

Table ACTIVITY	Activity information *Fields:*	
**	CODE	Activity code
	BUDGET_COST	Budget cost? (Y/N)
	OTHER_COST	Other cost? (Y/N)
	FED_DEDUCT	Deduct cost from federal return? (Y/N)
	UNITS	Units of each activity
	NAME	Description of the activity
Table OUTPUT	Output information *Fields:*	
**	CODE	Output code
	FED_RETURN	Return to federal govt.? (Y/N)
	OTHER_RETURN	Other return? (Y/N)
	UNITS	Units of each output
	NAME	Description of the output
Table TY	Treatment-type information *Fields:*	
**	CODE	Treatment-type codes
	NAME:	Treatment-type description
Table THEME	Level names *Fields:*	
**	NO	Level number (1-8)
	NAME	Name of each level
Table ID	Level-identifier names *Fields:*	
**	THEME_NO	Level-identifier number
**	CODE	Two-digit identifier code
	NAME	Six-digit identifier name
	DESCRIPTION	65-character description
Table YRG	Temporal information *Fields:*	
**	NO	Sequential year group number
	BEGIN_YEAR	Beginning year, such as 1993
	LEN	Length, in years, of each year group
	PERIOD	Period a year group is in
Table TRAIT	Qualifier and Age information *Fields:*	

**	CODE	Qualifier code
**	TYPE	Type of qualifier, A for age, Q for any other type of qualifier.
	Description	Description of each qualifier
Table AA	Analysis area, resource project, and road project information *Fields:*	
**	CODE	Analysis area or project code
	TYPE	A for analysis area, R for road project.
	AMOUNT	Acreage of analysis area
Table AA_ID	Analysis area codes and level-identifier information. *Fields:*	
**	AA_CODE	Analysis area code
**	THEME_NO	Level-identifier number
**	ID_CODE	Level-identifier code
Table ZONE	zone information *Fields:*	
**	CODE	Zone code
	DESCRIPTION	Zone description
	AMOUNT	Acreage of the allocation zone
Table ZN_AA	Analysis area acreage in each zone *Fields:*	
**	ZN_CODE	Zone code
**	AA_CODE	Analysis area code
	AMOUNT	Acreage of the analysis area in the zone
Table ZN_COL	Prorated prescriptions for AA in each Zone *Fields:*	
**	COL_TYPE	Type of column (zone)
**	COL_NAME	Name of the zone prescription
**	ZN_CODE	Zone name
	PRORATION	Proration proportion
Table CAC	Coordinated allocation choice (CAC) information *Fields:*	
**	CODE	CAC code
	DESCRIPTION	Description of the CAC
	ZN_CODE	Zone code a CAC is themed to
Table RX	Prescription code information *Fields:*	
**	CODE	Two-character prescription codes
Table RX_ID	More Prescription code information *Fields:*	

	**	RX_CODE	Two-character prescription code
	**	THEME_NO	Level-identifier number (1-8)
	**	ID_CODE	Two-character level identifier code
Table AO_OCCUR		Activity and output amounts occurring with a prescription *Fields:*	
		CODE	Activity or output code
		COL_TYPE	Column type. A= analysis area
	**	COL_NAME	Column or prescription name
		TRAIT_CODE	Qualifier code
		TRAIT_TYPE	Qualifier type
		YRG_NO	Year group number
		TT_CODE	Treatment type code
		AMOUNT	Level of the activity or output in solu-tion
Table COLUMN		Column information *Fields:*	
	**	TYPE	Column type. 1 = prescription
	**	NAME	Column name
		AA_CODE	Analysis area associated with a column
		RX_CODE	Prescription associated with a column
		CAC_CODE	CAC associated with a column
		IMPL_PERIOD	Implementation period
		AMOUNT	Amount of the column in solution

After the data base files are created, F2P then creates a set of primary index (.PX) files. Primary index files are related to the key fields in the tables. They list the records in key-field order. Paradox keeps the tables sorted in this key-field order and prevents records with duplicate key fields from being loaded into the tables. Once these indexes are set up, Paradox maintains them and uses them automatically without user intervention. This version of F2P does not permit key fields (.PX files) on the RX_ID or RX tables.

E. LOADING THE DATA

After F2P has created the data base and primary index tables, it will load the data from the FORPLAN relational flat file into the data base tables. Each relational flat file should reside in a separate subdirectory, and F2P should be executed from those subdirectories. If data from a new flat file is to replace data in existing tables, you must first delete all data base (.DB) and primary key (.PX) files before F2P can create new tables and load the information. If you attempt to run F2P in a directory where the data base files already exist, you will get an error message and F2P will cease processing.

F. ACCESSING THE TABLES

Once F2P has been successfully executed you may access the solution information via the Paradox data base tables. Execute Paradox from the same directory containing the data base file to create your own custom solution reports.

G. GETTING ASSISTANCE

If you have any questions, suggestions, or problems with F2P, please contact K.SLEAVIN:W04A at (303) 498-1833.

APPENDIX E. SUPPLEMENTARY SOFTWARE

The increasing complexity of forest planning issues requires the development of advanced tools for precise quantification and analysis of resource values and tradeoffs. Region 6 has developed a variety of software tools to complement FORPLAN. These programs are stored in self-extracting compressed files and have an ".EXE" extension. These programs are used in R6 with the relational data base program Paradox, but can also be used with ORACLE or any other data base management software. To obtain the software and documentation

1) Use the DG retrieval process (RIS) to move the files of interest to your DG. The files are located in

   :R06A:STAFF:PLAN:MERZ:PROGRAMS:

2) Transfer the files to your microcomputer.

3) From the MS DOS command line, type the desired file name. The programs and associated data base files will be loaded.

A. BUILDING AND INTERPRETING FOREST DATA SETS

FROGS.EXE: Building and Interpreting FORPLAN data sets.

Two programs are available to interpret and build FORPLAN data sets. These programs consolidate prior utilities developed in Region 6. They are contained in the self-extracting compressed file FROGS.EXE. Their names and functions are

FTRANS.EXE Translates FORPLAN data for importation to data bases.
FBUILD.EXE Converts ASCII files from data bases to the FORPLAN format.

These programs translate or build the following eight sections of the FORPLAN input.

1)	P2.1 and P2.2	IDENTIFIERS AND IDENTIFIER AGGREGATES
2)	P3.4 AND P4.4	COMMON COSTS AND RETURNS
3)	P5.1 - P5.8	ALLOCATION ZONE DATA
4)	P7.6	ABSOLUTE CONSTRAINTS
5)	P8.4	AP PRESCRIPTION SOURCE
6)	P8.6	HARVEST SOURCE
7)	P9.31	ANALYSIS AREA DATA
8)	P9.1	YIELD TABLE DATA

A brief description of the process used to translate each data section follows:

The "IDENTIFIERS" and "IDENTIFIER AGGREGATE" data are written to the file IDDATA. This file contains a data record for each raw level identifier and each combination of a raw and aggregate identifier. With these data one can unravel the mystery of how aggregates are themed in a FORPLAN model.

In FORPLAN "COMMON COSTS FOR ACTIVITIES" and "COMMON RETURNS ON OUTPUTS" are two separate data sections. FTRANS writes these data sections to one file named CRDATA. The first field identifies whether the record is a cost or return.

Seven files describe "ALLOCATION ZONE" data:

ZONAA	Analysis area acreage (P5.1)
ZPARXPA	Prescriptions without timing (P5.2)
ZONES	Zone codes, names, and acres (P5.3)
ZXALC	Zone allocation choices data (P5.4)

ZOXAA	Zone by AA acreage data from the *ZONE data (P5.3)
ZAAMS	Zone by AA by Mgt Strategy (Level 7) data (P5.8)
ZTHEM	Zone by "themed" aa identifier by Mgt Strategy data (P5.8)

By cross-referencing the zone files, data errors (i.e., acres not adding up) can be identified and corrected. These files are also useful for converting a model to one that includes (or does not include) zones.

FORPLAN "ABSOLUTE CONSTRAINT" data are written to ACDATA. In hierarchical analysis, "short term" (1-3 decade) models determine harvest capabilities for sub-forest areas. The potential harvest acreage per treatment type, working group, sub-area, etc. is then aggregated and incorporated as constraints into 15 decade "long-term" models. These problems enable one to quickly move ABSOLUTE CONSTRAINT data sets into and out of existing FORPLAN models.

The "AP PRESCRIPTION SOURCE" data are written to APDATA while "HARVEST SOURCE" data are written to HSDATA. Link these files with the "IDENTIFIERS" to interpret the aggregate identifiers used in theming.

The "ANALYSIS AREA" (p9.31) data is written to AADATA.TXT. If a model includes prescription controls these data are written to file AAPCON.TXT.

All data contained in the FORPLAN "YIELD.FIL" are described with three data bases:

AYHEAD contains header (I CARD) data for each table;
AYDATA contains the yields, and
AYTTRT contains treatment types by period for time-dependent yields.

Further instructions on running these programs and processing the data are contained in the file **FTRANS.DOC** (included in FROGS.EXE).

B. OTHER SOFTWARE

Brief description of two other software packages of interest follow:

PKZ110.EXE

This routine is a file compression routine needed to download the FORPLAN and utility programs. Put this file in a separate subdirectory and type in PKZ110 from the MS DOS command line. The PKZIP, PKUNZIP and other accessory programs and documentation files will be automatically downloaded. Include PKZIP in your "path" statement. The Forest Service has purchased a site license for PKZIP, so you do not have to pay for it.

The BRIEF Editor

FORPLAN data sets can be edited using a variety of editing software. Below are directions on editing your data file using the "BRIEF" editing software package. These directions will enable you to look in the matrix print file (MAT.FIL) for errors, reference the format guide for the correct format, and correct your data set. This process may all be done with one execution of BRIEF.

1) First place a copy of the "FORPLAN Version 2: Input Formats" document on your microcomputer.
   - — RIS a copy of the document from the LMP INFO center to your Data General.
   - — Move the document, using CEO Connect, to your micro computer.

2) Execute the BRIEF software package from the subdirectory containing MAT.FIL:

C:\FORPLAN\TEST> B

3) Enter the name of the matrix file:

File to edit: MAT.FIL

4) The matrix file covers the whole screen, but you will need to split the screen into three windows, one for the matrix file, one for the data file, and one for the format document.
   - — press the **"F3"** function key
   - — press the **"left arrow"** key to select a side
   - — press the **"F3"** function key again
   - — press the **"up arrow"** key again

Now your screen is split into three separate windows, each with a duplicate copy of the matrix file on it.

5) Place the format guide in the upper left-hand window.
   - — press **"ALT-E"** keys
   - — type in the path name to the format guide

6) Place the data set in the right-hand-side window.
   - — press the **"F1"** function key
   - — press the **"right arrow"** key to move to the right window
   - — press the **"ALT-E"** keys
   - — type in the data file name.

7) You now have three windows, one with the data set, one with MAT.FIL, and one with the input formats document. Move to the window with the matrix file (F1 - left arrow - F1 - down arrow) and scroll through it to find errors. Use the format guide (top left screen) to find the correct format for data input, and edit the data set (right window) to correct the data.

8) The three windows are rather small and may be difficult to read or edit. We suggest that you "zoom" the window you are the most interested in. To "zoom" a window, place your cursor in the appropriate window (using the F1 and arrow keys) and type:

   ALT-F2

   The entire screen will be filled with the "zoomed" window. To "unzoom" a window, simply type: **ALT-F2** again. You can now switch between windows as you need to, and zoom windows in and out at will. The matrix file can be searched for errors, the format guide referenced for the correct data positioning, and the data file corrected.

9) Below is a summary of the window commands:

F1	Change windows
F2	Resize windows
F3	Create windows
F4	Delete windows
SHIFT-arrow	Switch windows
ALT-F2	Zoom windows

10) To search for errors in the matrix file, simply toggle "Regular Expressions" off (Ctrl-F6) and Search Forward (F5) for *E* and continue searching for additional occurrences with Search Again (SH-F5).

11) To exit BRIEF and save your alterations, type: **ALT-X**, then **W**, or type: **CTRL-X**.

Abstract

Kent, Brian M.; Bevers, Michael; Sleavin, Katherine E.; Merzenich, James P.; Heiner, Jill; Turner, Matt; Hall, Sarah B. 1992. Operations Guide for FORPLAN on Microcomputers (Release 14.2). Gen. Tech. Rpt. RM-235. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Rocky Mountain Forest and Range Experiment Station. 88 p.

This guide covers a number of topics related to running FORPLAN (Release 14.2) on microcomputers. These include installing the FORPLAN software, preparing batch files for FORPLAN, LINDO and C-WHIZ program execution, generating FORPLAN linear programming (LP) matrices, using the LINDO and C-WHIZ LP solution software, and generating FORPLAN solution reports.

Keywords: land management planning, linear programming, national forests, FORPLAN

Rocky Mountains

U.S. Department of Agriculture
Forest Service

Rocky Mountain Forest and Range Experiment Station

The Rocky Mountain Station is one of eight regional experiment stations, plus the Forest Products Laboratory and the Washington Office Staff, that make up the Forest Service research organization.

Southwest

RESEARCH FOCUS

Research programs at the Rocky Mountain Station are coordinated with area universities and with other institutions. Many studies are conducted on a cooperative basis to accelerate solutions to problems involving range, water, wildlife and fish habitat, human and community development, timber, recreation, protection, and multiresource evaluation.

RESEARCH LOCATIONS

Research Work Units of the Rocky Mountain Station are operated in cooperation with universities in the following cities:

Great Plains

Albuquerque, New Mexico
Flagstaff, Arizona
Fort Collins, Colorado*
Laramie, Wyoming
Lincoln, Nebraska
Rapid City, South Dakota

*Station Headquarters: 240 W. Prospect Rd., Fort Collins, CO 80526